12-26-2014

Approximate Dynamic Programming for Military Resource Allocation

Carl R. Parson

APPROXIMATE DYNAMIC PROGRAMMING FOR MILITARY RESOURCE ALLOCATION

DISSERTATION

Carl R. Parson,

AFIT-ENS-DS-14-D-16

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This is an academic work and should not be used to imply or infer actual mission capability or limitations.

AFIT-ENS-DS-14-D-16

APPROXIMATE DYNAMIC PROGRAMMING FOR MILITARY RESOURCE ALLOCATION

DISSERTATION

Presented to the Faculty
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy (Operations Research)

Carl R. Parson, B.S., M.B.A., M.S.

December 2014

AFIT-ENS-DS-14-D-16

APPROXIMATE DYNAMIC PROGRAMMING FOR MILITARY RESOURCE ALLOCATION

Carl R. Parson, B.S., M.B.A., M.S.

Approved:

//signed//	18 November 2014
Darryl K. Ahner, PhD, PE (Chairman)	Date
//signed//	18 November 2014
Richard F. Deckro, DBA (Member)	Date
//signed//	18 November 2014
Meir Pachter, PhD (Member)	Date
//signed//	18 November 2014
Lt Col Matthew J.D. Robbins, PhD (Member)	Date

Accepted:

Adedeji B. Badiru, PhD, PE, PMP, FIIE Dean, Graduate School of Engineering and Management	Date

AFIT-ENS-DS-14-D-16

Abstract

This research considers the optimal allocation of weapons to a collection of targets with the objective of maximizing the value of destroyed targets. The weapon-target assignment (WTA) problem is a classic non-linear combinatorial optimization problem with an extensive history in operations research literature. The dynamic weapon target assignment (DWTA) problem aims to assign weapons optimally over time using the information gained to improve the outcome of their engagements. This research investigates various formulations of the DWTA problem and develops algorithms for their solution. First, a two stage stochastic WTA problem is explored which assumes independence of the two stages. Next a two stage shoot-look-shoot (SLS) formulation is explored in which the second stage targets are dependent on the first stage allocations. A novel multi-stage DWTA formulation is then presented in which kill probabilities are dynamic and dependent on the current set of targets. Finally, an embedded optimization problem is introduced in which optimization of the multi-stage DWTA is used to determine optimal weaponeering of aircraft.

Because of its flexibility and applicability to sequential optimization problems, approximate dynamic programming is applied to the various formulations of the WTA problem. Like many in the field of combinatorial optimization, the DWTA problem suffers from the curses of dimensionality and optimality is often computationally intractability. As such, approximations are developed which exploit the special structure of the problem and allow for efficient convergence to high-quality local optima. Finally, a genetic algorithm solution framework is developed to test the embedded optimization problem for aircraft weaponeering.

To my wife and children; Mom, Poppa, and Grammy.

Acknowledgements

I would like to express my sincere appreciation to my research advisor Dr. Darryl Ahner for his extensive help in the development of this dissertation. I would also like to thank my research committee, Dr. Richard Deckro, Dr. Meir Pachter, and Lt Col J.D. Robbins, PhD, for their time and effort, which improved the quality of this research. Further, I would like to thank the analysts from AFRL who helped define this exciting problem, and specifically Ms. Teresa Dailey who provided our initial sponsorship. Finally, I would like to thank the faculty, staff, and students of AFIT who enhanced my professional, personal, and academic growth over the past 39 months.

Carl R. Parson

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

Table Page

APPROXIMATE DYNAMIC PROGRAMMING
FOR MILITARY RESOURCE ALLOCATION

I. Introduction

1.1 Background

The weapon-target assignment (WTA) problem is a classic resource allocation problem in the field of military operations research where the objective is to optimally assign M weapons to N targets such that the expected remaining target value is minimized (or total expected destroyed target value is maximized). Because of its applicability to numerous issues facing military analysts, such as ballistic missile defense, air-to-ground operations, and integrated air defense systems (IADS), this problem continues to be of significant operational importance. Additionally, because of the variety of formulations and the extreme complexity of each, the WTA problem is also significant in the theoretic realm.

The WTA problem was first formally posed in 1958 [63], and is known to be NP-complete [60]. Since then, much research has been done which provides exact (optimal) or heuristic (not provably optimal) solutions for a variety of instances of the WTA problem.

Though it can be found under many names, two specific types of WTA problem are found: *static* and *dynamic*. In the static WTA (SWTA) problem all information is known *a priori* and all allocations are made at one time. The dynamic WTA (DWTA) problem may take many forms, though the underlying structure of each of

these is a sequential decision process. In the DWTA, at stage t, weapons allocations must be made, the outcome of which impacts the future state space.

In both cases (SWTA and DWTA), there is a single-shot probability of kill for a given weapon-target assignment. For the SWTA, the stochastic nature of the problem is handled using simple expectations of the outcomes. However, for many of the DWTA formulations, future stages present an additional stochastic element where the variance of each outcome significantly impacts future decisions. As such, the DWTA maintains increased complexity for which few solution techniques exist.

For deterministic problems with static resources and requirements, numerous methods exist for efficient search of the solution space. There are several cases where optimality has been proven, each under simplifying assumptions. As many practical problems are stochastic and dynamic in nature, most traditional methods fall short. Additionally, as the number of weapons and targets increase, the state, decision, and outcome spaces within a dynamic programming framework increase exponentially. These are known as dynamic programming's curses of dimensionality [82]. Much of the existing research focuses on solution techniques for the static problem in lieu of the more complex, and practical, dynamic formulation. Therefore, it is important to develop methodologies which can handle this sequential decision process efficiently while still providing high-quality solutions.

1.2 Motivation

Analysts at the Air Force Research Laboratory (AFRL) are developing a simulation framework in which future weapons concepts may be tested prior to development. As part of their framework they must analyze the effect a specific mix of weapons may have against specific IADS scenarios. Their current methodology steps forward in time, randomly selecting a weapons employment strategy until the aircraft is de-

stroyed. At this point, the simulation steps back to the last time the aircraft was alive and tries a different tactic. This process is repeated until the aircraft makes it through the whole simulation and the full policy is recorded as a possible solution. A set of candidate solutions are then selected, simulated repeatedly, and statistics are collected. This methodology is generally inefficient, especially given the sequential nature and complexity of the embedded assignment problem.

The objective of the AFRL research effort is to optimize a mix of weapons to inform acquisition of future systems while examining any synergistic effects kinetic and directed energy weapons may have together. Because the target set is assumed to be an IADS, weapons' capabilities will likely change as targets are destroyed. Currently, there is no formulation in the literature that considers probabilities of kill which evolve as a function of the target set. Additionally, within the simulation, weapon assignments should consider their impact on the evolution of the system, instead of being myopically allocated. Because of this, a dynamic instance of the weapon target assignment is appropriate.

To optimize the set of weapons used, a genetic algorithm (GA) has been developed for which an objective (or fitness) value must be computed for each design point. For this problem, the fitness value depends on the allocation and capability of each weapon being investigated. Few efficient allocation strategies are present in the literature, and where they exist, they are for static assignment. Further, within the GA, no methodology is in place to define when or how the weapons are to be fired. The sequence of how the weapons are fired may be considered in the design space, impacting the size of the space to be searched. As an alternative, using the sequential solution nature of dynamic programming, we can more efficiently search the design space, by providing the optimal allocations to the simulation. Dynamic programming has the flexibility to be integrated directly within the simulation by

yielding an efficient policy through a functional approximation given the state of the system.

Because of the many complexities of the motivating problem, both a theoretical advancement of provable optimality and practical application are necessary. Further, gaps in the current literature must be addressed which consider dynamic kill probabilities, the large decision space of the DWTA problem, and the embedded nature of the GA solution.

1.3 Research Contributions

Though it is a classic resource allocation problem, the weapon-target assignment problem is still of interest to military practitioners and academics alike. This dissertation develops numerous solution techniques for various formulations of the DWTA problem. Specifically, this research provides the following contributions:

- Develop an adaptive dynamic programming algorithm which optimally solves a two-stage stochastic WTA problem with homogenous weapons
- Extend the adaptive dynamic programming method to a shoot-look-shoot (SLS) DWTA problem to efficiently provide high-quality solutions
- Formally pose the cooperative, multi-stage, dynamic weapon-target assignment problem
- Use of order statistics to reduce the size of the allowable decision space within a dynamic programming solution methodology
- Formulate and solve an embedded optimization problem in which the sequential allocation of weapons to targets determines item utility within a knapsack problem

- Develop a genetic algorithm solution framework which integrates the use of ADP to determine optimal weapons allocations for testing within a simulation

1.4 Paper Structure

The remainder of this dissertation is organized into six chapters. Chapter II contains a detailed literature review. This incorporates both a survey of the weapon target assignment problem, followed by a discussion of approximate dynamic programming as a solution methodology. Chapter III provides an optimal method for a two-stage stochastic WTA problem. Chapter IV extends the research of Chapter III and investigates a two stage shoot-look-shoot formulation of the WTA problem where the second stage targets depend on the outcome of the first-stage assignments. Chapter V develops an approximate value iteration methodology through the use of order statistics, and Chapter VI describes the case study in which the solution methodologies are integrated within a general simulation framework that solves a complex embedded optimization problem. Finally, Chapter VII provides conclusions, highlights the major contributions, and provides recommendations for future research.

II. Literature Review

2.1 Weapon-Target Assignment Problem

The weapon-target assignment (WTA) problem is a well known military operations research problem. Though the static WTA was initially posed formally by Manne [63] as a special case of the transportation problem, it was first informally posed by Merrill Flood at The Princeton University Conference on Linear Programming in March of 1957 as similar to the personnel assignment problem [66]. Another item of interest for the WTA problem as shown in [63], is that Dantzig is responsible for the formulation that is widely used today. Since this time, substantial research has been dedicated to determine the optimal allocation of weapons to targets. Two general formulations are investigated in literature: *static* and *dynamic*. In the static formulation, though the outcomes of the assignments are stochastic, all information is assumed known prior to making the assignment, and all allocations are made at one time. This is the problem posed by Manne [63]. First formulated by Hosein, Walton and Athans [48], the dynamic problem has similar stochastic elements as the static problem, but assignments are made in multiple stages. Likely due to the standardized formulation of the problem, the static WTA (SWTA) problem is the most widely researched formulation in the literature. An early extension of the problem is given by Day [28] who uses a three-stage decomposition technique to solve a weapons allocation problem by relating the assignment problem to that of decentralized planning in large organizations. Matlin [66] provides the first survey of missile allocation literature, which is later updated by Cai *et al.* [42]. Eckler and Burr [31] also provide numerous examples and mathematical models of missile allocation and target coverage problems. The WTA problem is equivalently postured as both offensive, where the objective is to maximize the damage to the targets, and defensive, where the

objective is to minimize the value of any remaining targets. Other formulations also consider an asset-based defense, where the objective is to minimize damage done to a set of assets by assigning interceptors to incoming adversarial missiles [16][102][101].

2.1.1 Static Weapon-Target Assignment Problem.

The SWTA is formulated as follows. Let V_j denote the value of the j^{th} target, W_i denote the number of available weapons of type i. It is assumed that there are m weapon types and n targets. Let p_{ij} be the single shot probability of the i^{th} weapon killing the j^{th} target, such that the single shot probability of survival is $q_{ij} = 1 - p_{ij}$. The decision variable x_{ij} is the number of weapons of type i assigned to target j. The defensive SWTA problem is then formulated as a nonlinear integer program:

$$\min \sum_{j=1}^{n} V_j (\prod_{i=1}^{m} q_{ij}^{x_{ij}}) \tag{2.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \leq W_i \text{ for all } i = 1, 2, \ldots m, \tag{2.2}$$

$$x_{ij} \geq 0 \text{ and integer, for all } i = 1, 2, \ldots m, j = 1, 2, \ldots n. \tag{2.3}$$

The SWTA was shown to be *NP-complete* in 1986 by Lloyd and Witsenhausen [60]. As such, much research has been done in the past several decades to efficiently determine optimal solution methods. Two optimal solutions exist for simplifying assumptions of the SWTA. First, given a homogeneous weapon set, $p_{ij} = p_j$ for all i, denBroeder [30] shows optimality is achieved by evenly distributing the weapons across as many targets as possible using the maximum marginal return (MMR) algorithm. This algorithm assigns weapons sequentially to the weapon with the highest

remaining expected damage value until all weapons have been allocated. The second instance assumes that each target can have at most one weapon assigned to it [24][74].

2.1.2 Current Literature of the Static Weapon- Target Assignment Problem.

Considering any one specific formulation, the majority of the literature has been dedicated to efficiently solving the SWTA problem formulation; in addition, several papers have been developed since the 2006 survey by Cai *et al.* [42]. As with many NP-complete or other combinatorial optimization problems, the existing literature applies a wide variety of methods to quickly generate high-quality, but generally suboptimal, solutions. Ahuja *et al.* [5] present commonly cited results and give a benchmark for solution quality through lower bounding (for the minimization problem) techniques. Their formulation uses integer linear programming and a general integer network flow problem using a minimum cost flow to determine a new lower bound (if minimizing). The authors also provide a very large-scale neighborhood improvement heuristic algorithm which quickly solves moderately sized instances (up to 80 weapons and targets) optimally while providing high-quality solutions for larger problems (up to 200 weapons and targets). As previously discussed, the earliest optimal methods were presented by denBroeder [30] under a homogenous weapon set assumption, known as the MMR algorithm. This greedy method is also a fast method for bounding of the solution when the homogeneous weapons assumption has been relaxed. Chang *et al.* [24], and Orlin [74] developed optimal methods under the assumption that each target can have no more than one weapon assigned to it. These methods exploit the underlying network flow structure of the SWTA problem.

Since the first approximation technique for the SWTA was done in 1966 [28], a gamut of popular metaheuristics have been applied to the SWTA problem. This in-

cludes ant colony optimization (ACO) [57][88], particle swarm [34] [104] (of a slightly more generalized resource allocation problem), and genetic algorithms (GAs) [19] [58] [49] [61]. As stand-alone methods, simulated annealing (SA) and tabu search are two popular heuristics for which literature gaps appear to exist. There are, however, hybrid methods used to provide solutions for the SWTA, to include ACO with SA [97], GA with ACO [33], GA using greedy search procedures to improve the quality of the offspring [59], and particle swarm with embedded greedy algorithms [50]. Turan [95] provides a comparison of several heuristic algorithms for the WTA problem and poses a new hybrid algorithm consisting of particle swarm and random search to produce higher-quality solutions. In addition to these popular metaheuristic methods, several other approximation methods have been used for the SWTA. Chen, Ren, and Deng [26] use a modified MMR type algorithm after changing the network representation from a one-to-many to a one-to-one mapping to efficiently approximate the optimal value. Rosenberger *et al.* [85] compares the sequential application of the auction algorithm in a greedy fashion to an exact (but computationally expensive) branching and bounding technique. Sahin and Leblebicioglu [62] apply fuzzy reasoning to approximate optimum allocations in real-time for use on a battlefield. Lastly, Lagrangian relaxation [72] was used to decompose the problem into two tractable subproblems while iteratively updating the Lagrange multipliers. Of the extensive amount of research done for the SWTA, Ahuja *et al.* [5] appears to be the most widely accepted solution which solves the general SWTA problem. Next, the more complex dynamic weapon target assignment formulation is discussed, followed by a review of existing literature.

2.1.3 Dynamic Weapon-Target Assignment Problem.

The DWTA divides the total duration of an offensive attack into several discrete time steps in which information is obtained about the allocation outcomes of the previous stages. Any targets destroyed during a stage are no longer targeted in subsequent stages, allowing the operator to make better use of their weapons. The basic assumptions of the DWTA, as outlined in [47], are as follows:

- In each stage, a subset of weapons is selected and committed simultaneously.
- The outcomes of each stage are observed prior to the following stage (this can either be perfect knowledge or stochastic, though Hosein [47] assumes perfect knowledge)

Furthermore, Hosein and Athans [47] methodology obtains solutions by

- Re-solving the problem at each stage using previous stage information
- Computing the optimal assignment for the current stage always assumes optimal assignments will be made in subsequent stages
- Selecting weapons at each stage with the goal of optimizing the expected sum of realized values over all stages

The multi-stage problem as formulated in [48] is as follows. Let $T \triangleq$ the number of time stages, $M \triangleq$ the number of weapons, $N \triangleq$ the number of targets, and $V_i \triangleq$ the value of target i for $i = 1, 2, \ldots, N$. Let $p_{ij}(t) \triangleq$ the single-shot probability of kill if weapon i is assigned to target j in stage t, $i = 1, 2, \ldots M$, $j = 1, 2, \ldots N$, $t = 1, 2, \ldots T$, and $q_{ij}(t) = 1 - p_{ij}(t)$ be the corresponding probability of survival. Define the decision variables x_{ij} as

$$x_{ij} = \begin{cases} 1, \text{ if weapon } i \text{ is assigned to target } j \text{ in stage 1} \\ 0, \text{ otherwise} \end{cases}$$

Next, define the N-dimensional binary vector target state $\mathbf{u} \in \{0,1\}^N$ and the M-dimensional binary vector weapon state $\mathbf{w} \in \{0,1\}^N$, where

$$u_j = \begin{cases} 1, \text{ if target } j \text{ survives stage 1} \\ 0, \text{ if target } j \text{ is destroyed in stage 1} \end{cases}$$

and

$$w_i = \begin{cases} 1, \text{ if weapon } i \text{ is not used in stage 1} \\ 0, \text{ if weapon } i \text{ is used in stage 1} \end{cases}$$

Then, for any initial weapon-target assignment, x_{ij}, $\mathbf{u}$ is an N-dimensional random vector at the start of the second stage which captures the outcomes of the assignments. As shown in [47], the distribution of the u_j's is

$$P[u_j = k] = k \prod_{i=1}^{M} (1 - p_{ij}(1))^{x_{ij}} + (1-k) \left\{ 1 - \prod_{i=1}^{M} (1 - p_{ij}(1))^{x_{ij}} \right\} \tag{2.4}$$

for $k = 0, 1$ and $j = 1, 2, \ldots, N$. Equation 2.4 determines the probability with which states transition over time. The weapon states then transition over time using

$$w_i = 1 - \sum_{j=1}^{N} x_{ij}, \qquad i = 1, 2, \ldots M. \tag{2.5}$$

Define $F_2^*(\mathbf{u}, \mathbf{w})$ as the optimal cost of a $T-1$ stage problem given an initial state $(\mathbf{u}, \mathbf{w})$, then, because this is defined in terms of T two-stage subproblems, by recursively using

$$F_{T+1}^*(\mathbf{u}, \mathbf{w}) = \sum_{j=1}^{N} V_j u_j, \tag{2.6}$$

the DWTA formulation is:

$$\min_{x_{ij}} F_1 = \sum_{\omega \in \{0,1\}^N} P[\mathbf{u} = \mathbf{w}] F_2^*(\omega, \mathbf{w}) \tag{2.7}$$

subject to

$$x_{ij} \in \{0, 1\}, \qquad i = 1, 2, \ldots M, \qquad j = 1, 2, \ldots N, \tag{2.8}$$

$$\text{with } w_i = 1 - \sum_{j=1}^{N} x_{ij} \tag{2.9}$$

Here, ω is the random outcome based on the current stage assignment. In words, the objective is to minimize over all possible second stage target states to determine our optimal expected second stage return. Recursion is used $T-1$ times to determine the optimal T-stage weapon-target allocation.

Hosein [47] also provides a list of important properties of the DWTA and that, given these properties, obtaining algorithms which efficiently solve this problem optimally is unlikely. These characteristics are:

(a) Dynamic WTA problem is *NP-Complete*

(b) DWTA is discrete (and integer) - fractional weapon assignments are not allowed

(c) Dynamic (and sequential) - the current stage outcomes inform future decisions

(d) Nonlinear - with a convex objective function

(e) Stochastic - the outcomes of the assignments are probabilistic in nature

(f) Large-Scale - as problem size increases, enumeration techniques become impractical or computationally intractable

If the state is defined such that it consists of the number of stages remaining, and the current weapon and target states (based on the transitions already defined), then,

structurally, all the elements necessary to classify it as a sequential decision process are present. Additionally, since the current decision only depends on the current state (which captures the necessary previous information to make our decision), the Markovian property is satisfied as well. As such, this problem is ideally suited for solution using dynamic programming. However, as is often the case for dynamic programming problems, as the state space increases, so does the decision space, as well as the possible outcomes at each stage. Therefore, this problem suffers from the three curses of dimensionality, and as the number of states increase (i.e. the number of weapons and targets increase), traditional solution methods become intractable. These three curses of dimensionality arise from exponential growth of either the state space, decision space, or outcome space, or their combined increase. Before moving on to a discussion of mitigation strategies for these curses, a simplified DWTA problem consisting of only two stages is presented.

2.1.4 Two-Stage DWTA.

As a simplification to the multi-stage problem posed by Hosein [48], Murphey [70] defined a two-stage stochastic programming model of the weapon target assignment problem. In this model, consider the probability that an adversary has a total stockpile of weapons and shoots a portion of them in the first stage, with the remainder of the weapons, known to a probability distribution, arriving in the second stage. An important distinction for this problem is that the second stage target arrivals are independent of the first stage assignments. Let n_1 targets arrive in stage 1, and n_2 targets arrive in stage 2. Let the random vector $\omega \in \Omega$ denote the number of targets in the second stage. Suppose the probabilities of survival, q_i, and target values, V_i,

for each target i are given. Then the 2-stage WTA programming formulation is:

$$Z_1(x) \quad = \min_x f_1(x) + E_{\omega\in\Omega}[Z_2(x,\omega^j)] \tag{2.10}$$

subject to

$$\sum_{i=1}^{n_1} x_i \leq M,$$

$$x \leq b$$

$$x_i \in \mathbb{N}^+, i = 1\ldots, N$$

where

$$f_1(x) = \sum_{i=1}^{n_1} V_1^i (q_1^i)^{x_i}$$

is the first stage value function of the first stage assignment x and is integer convex, $E_{\omega\in\Omega}[Z_2(x,\omega^j)]$ is the expected second stage value and is integer convex (meaning the relaxation problem is convex) where ω^j is a scenario in stage 2 and is solved using the MMR algorithm, $\sum_{i=1}^{n_1} x_i \leq M$ is the resource capacity constraint, and b is the vector denoting the maximum weapons that can be assigned to any one target.

$Z_2(x,\omega^j)$ is the solution to the second stage problem and is expressed as:

$$Z_2(x,\omega^j) = \min_y f_2^j(y) \tag{2.11}$$

$$\text{subject to } \sum_{i=1}^{n(i)} x_i + \sum_{i=1}^{n_2(\omega^j)} y_i = M,$$

$$y \leq b$$

$$y \in \mathbb{N}^{n_2}$$

where

$$f_2^j(y) = \sum_{i=1}^{n_2(\omega)} V_2^i(\omega)(q_2^i(\omega))^{y_i}$$

is the second stage value function. $f_2^j(y)$ depends on the outcome of ω and is integer-convex.

Murphey [70] uses a decomposition method to decouple the first and second stage. The decomposition method first solves a variant of the stage 1 problem called the current problem:

$$\min_{x,\theta} f_1(x) + \theta \tag{2.12}$$

subject to

$$\begin{aligned} Ax &\leq b, \\ x &\in X \end{aligned} \tag{2.13}$$

with a scalar θ taking the place of the second stage value so that

$$\theta \geq \sum_{j=1}^{s} p^j Z_2(x, \omega^j)$$

where s is the total number of scenarios and p^j is the probability of scenario j occurring.

Murphey [70] uses stochastic decomposition to come up with approximate solutions for this formulation.

2.1.5 Other Literature of the Dynamic Weapon- Target Assignment Problem.

Though it has not been researched to the extent of the SWTA problem, the DWTA problem provides a more practical implementation by including a temporal component. As such, the DWTA is a much more complex problem from a mathematical

standpoint and has received a fair amount of attention in the literature. Similar to the SWTA, numerous methods have been employed to provide solutions for various types of DWTA problems. As the originator of the dynamic instance, Hosein [47] provides several results which are generalizable to the DWTA problem. Murphey [70][71] uses stochastic decomposition for the two-stage problem previously defined. An extension of the generalized two-stage problem called the shoot-look-shoot target assignment problem also has a fair amount of associated literature, but will be discussed in the next section. Specific to the general DWTA problem, Chang [24] uses a static WTA approximation scheme within an iterative linear network flow framework to efficiently provide high-quality solutions for the DWTA. Because of the integrality constraint of the decision variables, the chromosome representation within a GA presents a useful scheme for solving both the static and dynamic versions of the WTA problem. As such, much work has developed hybrid GAs to assist in solving the DWTA. Wu *et al.* [99] apply a modified GA to the DWTA and introduces weapon use deadlines within the problem formulation. These deadlines follow the principles of scheduling theory, and are in the form of additional constraints such that a weapon has to be shot at a target by a specified time or it is rendered unusable. The authors call their method a modified GA because it applies a basic GA iteratively, assigning a weapon to a target (possibly suboptimally) immediately before the deadline is reached. Xin *et al.* [101] develop a heuristic which uses problem information (domain knowledge) and constraint programming to assign priorities to assignments. Evolutionary heuristics which use a hybridized GA with memetic algorithms have also been applied to the DWTA [25]. Additionally, Khosla [54] applies a hybrid heuristic which uses a simulated annealing (SA) heuristic to determine the fitness of a population within a GA framework. Other heuristic techniques applied to the DWTA include Tabu Search [102], ACO with tabu table updates [103], and a modified Hungarian method with

PSO [56]. Lastly, exact dynamic programming [91][89] has also been applied to the DWTA. The last portion of the WTA literature review focuses on the specific shoot-look-shoot scenario, as well as some miscellaneous WTA formulations and solution methods not explicitly for WTA fproblems.

2.1.6 Other Target Assignment / Weapons Allocation Literature.

Because of the numerous articles dedicated to it, methods for solving the specific shoot-look-shoot (SLS) problem, as well as some other miscellaneous allocation methods, are now discussed. The SLS problem is a dynamic weapon target assignment problem which allows for multiple allocation stages with some form of battle damage assessment after assignments are made. At the end of each stage, the outcomes of the allocations are known according to some probability distribution prior to making the subsequent stage allocations. The complexity of the SLS problem is the dependency of future stages' target sets on previous weapon assignments. The utility of the SLS problem is that it demonstrates the impact current outcomes have on initial weapons allocations with the knowledge that some weapons need to be kept for future stages. Additionally, in a multi-stage WTA formulation, a myopic SLS policy could be implemented at each stage to provide a bound on the solution.

Manor and Kress [64] prove optimality of a multi-stage greedy SLS solution against a homogeneous target set assuming imperfect damage information. They also show that the original SLS problem is equivalent to a finite horizon deteriorating bandit problem, which dynamically allocates a single resource amongst a fixed number of arms. Aviv and Kress [8] evaluate several SLS tactics (such as the persistent shooter, fixed bound on munitions and dynamic bound on munitions) and analyzes their efficiency when damage information is uncertain (or incomplete). Glazebrook and Washburn [35] provide a brief survey of, and further investigate the SLS problem

considering several scenarios in which information may be perfect or imperfect, the time horizon is finite or infinite, and homogeneity (or non-homogeneity) of weapons is considered. They approach the problem as a partially observable Markov decision process (POMDP), and apply dynamic programming citing *the computational intractability of their methods as problem size increases.* Yost and Washburn [105] also decompose the problem into a linear program to obtain an initial (bound) set of policies and use dynamic programming to help improve the policies. The dynamic programming subproblem is also viewed as a POMDP, as in [35]. Karasakal [51] applies integer programming decomposition to determine SLS policies for allocating surface-to-air missiles within a naval task group. Castañon [23] approaches the SLS problem as a two stage resource allocation where the goal is to maximize the first stage allocations while considering the second stage recourse requirements. The formulation then takes on a similar form to that of the two-stage stochastic control problem defined by Murphey [71], and is similar to a constrained two-stage form of Bellman's equation [10]. Linear interpolation and Lagrangian decomposition are then used to quickly approximate recourse actions (the 2nd shooting stage in the SLS). These values are then used recursively to greedily solve the first stage problem.

Lastly, there are several other refereed journal articles which focus on areas related to the WTA assignment problem, but are generally not classified as such. For brevity, and because of their uniqueness, each paper and the methodology used is introduced, but it is left to the reader to determine their specific formulation. Most of these papers have to do with interceptor allocation specific to ballistic missile defense (BMD) applications. Gorfinkel [40] uses decision theory to maximize the probability that a warhead is hit given that it is concealed in a cloud of decoys. Bracken, Falk, and Miercort [17] extend a model introduced by Phipps [78] in which two players exchange nuclear weapons. In this model, one player's remaining weapons are impacted

by the other player's strike package, thus the constraints of the second player become a function of the allocation of the first striker. This max-min problem is shown to be separable, but nonconvex, so traditional methods do not guarantee optimality via a saddle point. Instead, piecewise linear approximations are used, and solved via branch and bound. The linear approximations match the nonlinear objective function at a predefined number of gridpoints; as the number of gridpoints increases, the approximation becomes better, but at the cost of computation time. Metler, Preston, and Hofmann [68] present various solution techniques for five different defensive weapons allocation problems. The formulations investigated vary from a generalized DWTA problem, to a single threat-target assignment problem, while solution methodologies include linear and non-linear programming, branch and bound, greedy approximation and others. Wilkening [98] derives the size of defense necessary to meet defense objectives based on target kill probability, and applies it to national and theater missile defense. Bertsekas *et al.* [16] formulates the BMD problem as a Markov decision process (MDP) and uses neuro-dynamic programming where the cost-to-go functional approximation is achieved through neural network architectures. Brown *et al.* [18] apply a two-sided model to determine the optimal location to pre-position defensive platforms with the objective of minimizing the eventual damage from a ballistic missile attack. Menq *et al.* [67] uses discrete Markov decision process modeling as a means for providing distribution functions for BMD so that more accurate planning and cost analysis may be used in practical settings. Arslan, Marden, and Shamma [7] develop a game-theoretical formulation for vehicle-target assignment in which a set of vehicles cooperatively assign themselves to a set of targets to optimize some utility function.

2.2 Approximate Dynamic Programming

2.2.1 Dynamic Programming.

First, the the major concepts and assumptions which are used when considering dynamic programming as a solution methodology are briefly introduced. As a discipline, dynamic programming is a "collection of mathematical tools used to analyze sequential decision processes" [29]. As discussed in Denardo [29], regardless of how unrelated two different processes may seem, there are several underlying components common to all sequential decision processes. Specifically, at each decision epoch, the process is in some *state*, the goal is always (at least it should be) to make the best (or *optimal*) decision given the state that one is in, and finally, that based upon what decision is made at that given point, there will be an outcome that one will transition to via some sort of *functional, or transitional equation.* It is also assumed that the decision will either incur a *cost*, or the decision maker will obtain some immediate *reward* for making the decision. Once a transition takes place and the cost has been incurred, the decision maker will then be faced with an updated decision and the process will start over. One critical assumption for dynamic programming, however, is that once the transition has been made, what happened previously is entirely captured in the new state, thus future decisions do not depend on what happened in the past, and only depends on where the decision maker is at that epoch. This is also known as the *Markovian* property, without which much of the underlying mathematics would be substantially complex.

As previously discussed, for a generalized dynamic programming problem, several structural elements will always be present. Using the notational conventions of Bertsekas [12][13][15], they are defined as follows. Let k be the index of either a discrete time step, or a discrete decision epoch in continuous time. Then x_k is the state of the

system at k, and contains everything necessary to make a decision, u_k. In stochastic cases (discussed further later), there is a noise element, w_k, representing a random occurrence or outcome which may be based on the state, the decision, both, or neither. Finally, consider a time horizon N which tells the point at which to terminate recursion or it may represent the number of decision epochs. As will be discussed later, this horizon may be finite ($N < \infty$) or infinite $N = \infty$. The transition function for the stochastic formulation is of the following form:

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1 \tag{2.14}$$

where $f(\cdot)$ is a function defining the system dynamics. In dynamic programming, the costs (or rewards) are also assumed to be additive, meaning that at each decision epoch, the costs incurred up until the point are represented in some additive form, and will be added to future costs. The cost function will be some function of the state, decision and random outcome: $g_k(x_k, u_k, w_k)$. Therefore, given a terminal cost for being in the final state $g_N(x_N)$, the total cost over time is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \tag{2.15}$$

Additionally, in literature, the decision u_k may be represented as a function of the current state, or $u_k(x_k)$. Similarly, the random outcome may be a function of the state and the decision made, as in the case of weapons allocation, $w_k(x_k, u_k)$, but may also be itself a random occurrence (such as with the inventory demand example). However, to remain consistent with Bertsekas, this notation will not be used. In addition, a more explicit definition of these elements is found in [82] (pg 168).

Dynamic programming problems are solved using Bellman's equation [10] (extracted from [12])

$$J^*(x) = \min_{u \in U(x)} \underset{w}{E}\{g(x,u,w) + J^*(f(x,u,w))\} \tag{2.16}$$

This holds true when moving forward from any state in which the system may be. This is considered the principle of optimality, and (in words) states that there exists an optimal solution from any state to the end of the horizon. This is a powerful fact, and in many cases, this can be used to find all optimal paths (or decisions) from any state to any other set of states. One example of this is value iteration. During value iteration, each possible state is iterated over, and the optimal decision is determined. From these optimal decisions, transitions to a new state will occur, with which value iteration has provided the optimal decision.

2.2.2 Introduction.

Though the traditional methods for dynamic programming are very powerful, they fall victim to computational intractability as problem size increases; the curses of dimensionality. As such, means of mitigating this computational inefficiency must be examined. One such methodology is approximate dynamic programming. Powell [82] presents numerous examples of approximate dynamic programming for resource allocations problems. Because of the underlying sequential structure of the DWTA (and the possibility for selecting assignments in a sequential nature in the SWTA), small instances of the WTA problem may be solved using exact methods. However, as shown in the literature reviewed in Section 2.1, the tractability of these techniques decreases as problem size increases. Similarly, exact dynamic programming suffers from the same issues. Coined the three *curses of dimensionality*, for many solution methods (value iteration, policy iteration, and their variants), each state, decision, and (if present) possible outcome (random event or exogenous information process) need to be iterated over. As such, computational effort increases exponentially as

problem size increases. Several texts ([15][13][82]) are dedicated to presenting methods which address the curses of dimensionality of dynamic programs. Powell [82] states that all dynamic programs are able to be written in terms of a recursive relationship relating the expected cost (or reward) of being in a given state at a point in time, to the expected cost (or reward) of each possible future state. This relationship can make many problem sizes increase exponentially as a function of the state, decision, or outcome spaces.

A tutorial at the 2013 Industrial and Systems Engineering Research Conference given by Dan Adelman of The University of Chicago Booth School of Business [1] provides an insightful overview of the available approximate dynamic programming methods used to date. Figure 1 shows this hierarchy.

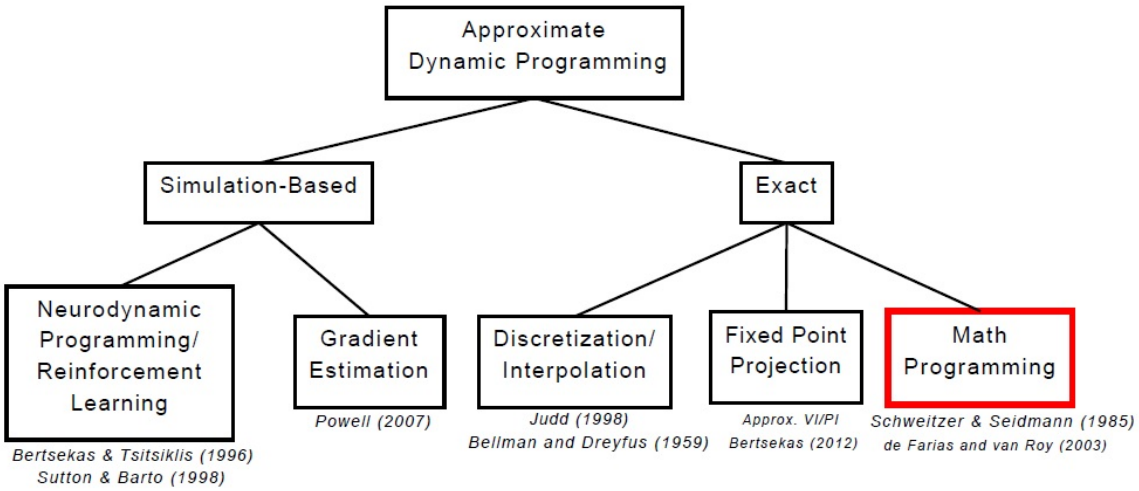


Figure 1. Approximate Dynamic Programming Methodologies

Though the state-of-the-art in approximate dynamic programming has made many advances since the dates provided in Figure 1, such as the 2nd edition of Powell's Approximate Dynamic Programming [82] and the 4th edition of Bertsekas' Dynamic Programming and Optimal Control [11] texts, Figure 1 gives a good breakdown from which to start. Specifically, it is differentiated between the exact and simulation-based

approximation methods. It is important to note that in each case, methods are used to approximate some portion of the problem, usually the expected future costs (or rewards), often called the "*cost-to-go*" or "*value*" function. This cost-to-go function is the $J^*(f(x,u,w))$ expressed in Equation 2.16. Whether this is done through some type of explicit mathematical programming method, or through Monte Carlo (or other) simulation, these techniques are designed to exploit the special structure of the specific problem to compute solutions which are nearly optimal, but are done using a fraction of the computational requirements. Powell [82] also provides a list of problems that must be addressed when trying to solve approximate dynamic programming problems in general:

- Forward dynamic programming avoids looping over all possible states, but still requires an explicit understanding of the one-step transition matrix and the possible states the system may transition to.
- The values obtained at each current state are known, with the need to know the values of the states which may be visited,
- Certain policies may cause the system to never visit states which, in the exact formulations, would net good solutions
- Each problem is unique, and while the approximate dynamic programming strategy is rather general, it cannot provide a mechanism for determining what will work best for the specific problem

Using these principles, an overview of some of the more widely used techniques in literature is no provided, with the goal of providing a sufficient spread while maintaining generality and conciseness. The focus is also restricted to the simulation-based techniques found in [13] and [82].

2.2.3 Lookup Tables and Q-Learning.

In some cases, a system may be so complex that it cannot be explicitly modeled mathematically. One example would be a complex simulation which needs to be optimized. The key to using Q-Learning is that the behaviors of the system are able to be observed directly, and controls are able to be placed on the system iteratively. Define for each state i (notation for the state is temporarily changed here to allow a more concise description of transitions) and decision u pair (i, u) for $u \in U(i)$, and the optimal Q-factor by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha J^*(j) \right). \tag{2.17}$$

For these problems, instead of approximating the cost-to-go function for the selected policy, at each iteration the Q-factors for each state are updated. This allows the multiple policy evaluation steps of policy iteration to be avoided. Instead, use value iteration on $Q_{k+1} = FQ_k$ defined by

$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{v \in U(j)} Q(j,v) \right), \qquad \forall (i,u). \tag{2.18}$$

Using this relationship, this is equivalent to a discounted Bellman equation, and the algorithm converges to Q^* from any starting point Q_0. In words, the Q factors are statistical estimates of the true future cost (or reward) given a state and action, which is beneficial because instead of needing an explicit transition function simulation outputs are used to iteratively update the estimates.

2.2.4 Approximate Value Iteration.

Next, consider an approach in which a model is known, but the specific one-step transition probabilities are unable to be determined. For this, Powell [82] suggests randomly generating a sample of K possible outcomes at each iteration of what may occur in the system (i.e. the random occurrence w_k) and select the probability that each of those randomly generated outcomes will occur. One such recommendation is to let $p_n(w_i) = \frac{1}{K}$ (here the probability is indexed by n, denoting the iteration for which the outcomes have been generated). The expected total costs are then approximated using the standard recursions of (2.16) using the generated outcome space. Next, the estimate of the value is updated using

$$J_n(x_n) = (1 - \alpha_{n-1})J_{n-1}(x_n) + \alpha_{n-1}\hat{v}_n \tag{2.19}$$

where $\hat{v}_n$ is the approximation discussed above. As will be seen in many of the applications of approximate dynamic programming, the stochastic smoothing equation (2.19 attempts to use observations of the inherently noisy data to approximate the moments of the actual distribution from which the observations are being drawn. Powell [82] provides extensive details on selection of step size, α, and a rigorous discussion of convergence properties for many instances.

2.2.5 Low-Dimensional Value Function Approximation.

The next method discussed concerns itself with reducing the dimensionality of the problem, by combining them into *aggregate states*. The effectiveness for this method in the context of approximate dynamic programming is that the aggregated states are used to determine the cost-to-go approximation, and at each iteration all states are iterated over [82]. In traditional methods, the aggregated states can also be enumerated over, but in many cases this leads to poor estimations of the problem

solution. Another way this method may help is by taking a continuous state space and discretizing it to use traditional methods. Next, the aggregation framework of Bertsekas [13] is introduced. Let $\mathcal{A}$ be a finite set of aggregate states, and define a disaggregation probability d_{xi} such that

$$\sum_{i=1}^{n} d_{xi} = 1, \qquad \forall x \in \mathcal{A} \tag{2.20}$$

where x is an aggregate state and i is the original system state. Then, for each aggregate state y and original system state j, the aggregation probability ϕ_{jy} is

$$\sum_{y \in \mathcal{A}} \phi_{jy} = 1, \qquad \forall j = 1, \ldots, n \tag{2.21}$$

Note that d_{xi} is essentially the proportion for which x is represented by i, and ϕ_{jy} is the "degree of membership of j in the aggregate state y." Define the matrices $\mathbf{D} = [\{d_{xi} | i = 1, \ldots, n\}]$ and $\mathbf{\Phi} = [\{\phi_{jy} | y \in \mathcal{A}\}]$. For clarity, these elements of the sets then represent the elements of their respective matrices. Then an approximation of Bellman's equation is obtained by $\mathbf{\Phi}\hat{\mathbf{R}}$ where

$$\hat{\mathbf{R}} = \mathbf{DT}(\mathbf{\Phi R}) \tag{2.22}$$

Here $\mathbf{T}$ is the recursion operator defined in Bertsekas [13].

One good example of this which should be applicable for this research is to consider a multi-stage weapon-target assignment problem formulated by Hosein [48]. At each time step, one example is to aggregate all future stages into a static weapon target assignment problem with the remaining weapons and targets.

2.2.6 Adaptive Estimation.

Adaptive estimation algorithms are broad and are also centered around the relationship shown in Equation (2.19). The primary idea is that we are trying to estimate a value $g(x)$ for being in state x, and $\hat{g}(x)$ is a somewhat randomized estimate of $g(x)$. A stochastic gradient algorithm then provides the result of Equation (2.19). There are many types of methods under Adaptive Estimation, such as recursive least squares, approximate value iteration, least squares temporal differences, and least squares policy evaluation. The use of these methods is in determining average costs of being in each state. For the purposes of this dissertation, each method is introduced with a short example of the type of problem they are applicable to. For further information, Powell [82] provides some insightful explanations of these giving closed form derivation using a single state. Another term (used by Bertsekas) for Adaptive Estimation algorithms is *Approximate Policy Iteration*.

Recursive Least Squares

Recursive least squares uses a means of generating approximations for the system using a linear combination of basis functions $\mathbf{\Theta}_f(x)$, where $f \in \mathcal{F}$ is considered a feature. The approximation is then

$$\tilde{J}(x) = \sum_{f \in \mathcal{F}} \beta_f \mathbf{\Theta}_f(x) = \mathbf{\Theta}(\mathbf{x})^T \beta \tag{2.23}$$

where β are traditional regression coefficients. These techniques are able to be applied any time the value function can successfully be approximated using linear regression. This art is left to the reader for a specified problem. Specific methods exist for cases where the analyst has stationary data, non-stationary data, and where multiple observations are obtainable.

Least Squares Temporal Differences

Identified by Powell [82] as one of the more powerful and attractive tools in approximate dynamic programming, using temporal differences provides a means of updating a functional approximation. At each iteration, estimates of the least squares regression coefficients β can then be updated. This method fixes a policy and then finds the best fit for the linear model. Additionally, the standard transition function is used to determine the next state to visit using this fixed. This is also known as on-policy learning [82]. The reason this method is so powerful is that it combines techniques which allow the user to obtain regression coefficient estimates and uses them in the traditional approximate dynamic programming solution framework.

Least Squares Policy Evaluation

Least squares policy evaluation uses basis functions developed for infinite horizon applications. At the n^{th} iteration, the regression coefficients are determined by

$$\beta_{\mathbf{n}} = \arg\max_{\beta} \sum_{i=1}^{n} \left(\sum_{f} \beta_f \mathbf{\Theta}_f(x_i) - (\hat{C}_i + \gamma \tilde{J}_{n-1}(x_{i+1})) \right)^2 \tag{2.24}$$

where $\hat{C}_i$ is a random variable providing the i^{th} contribution [to the value function] (this is considered a one-period contribution at the i^{th} step in the infinite horizon). Again, this method is just another way of determining the expected reward gained by being in state x to help compute average long-run rewards.

2.2.7 Issues of Simulation-Based Cost Approximation.

As discussed by Bertsekas [13], these methods primarily concern themselves with optimizing over an approximated single (or multi)-step lookahead approach. Determining these approximations is where the mathematics and art of dynamic programming merge. Getting an appropriate approximation can take both time and effort, and may not provide a robust methodology for solving problems which are closely

related to the original. Another issue arises with the statistical testing of the approximations, determining the rate of convergence, and solution quality. Each of the methods presented (and others found in the literature) have their benefits and drawbacks. Some may be the correct choice for the problem being investigated, and others may not be useful at all. The analyst has the task of generating an appropriate model (if available) and determining which solution technique(s) should be applied.

2.2.8 Approximate Dynamic Programming for Resource Allocation.

Several articles apply approximate dynamic programming for various resource allocation instances. This section is not intended to be a full literature review of these applications. Instead, the common themes amongst these papers are captured, and the feasibility of approximate dynamic programming as a solution technique for the WTA problem is developed. Powell has done a substantial amount of applied approximate dynamic programming work in resource allocation within the transportation industry [80][81][84][82]. One structural factor that is exploited is the declining marginal return of assigning an additional weapon to any given single target. As such, the value function is concave. Godfrey and Powell [37] have developed a method for approximating concave functions and have successfully applied it to a number of practical applications [80][94]. Castanon has also done work in approximate dynamic programming for resource control, to include sensor management [21], multiplatform path planning [77], and stochastic scheduling (along with Bertsekas) [14]. Another area which has a significant amount of literature is vehicle routing with stochastic demands [73][86][87][3]. Other resource allocations applications include activity networks for project planning [32][93], model predictive control [22], and high-dimensional generalized resource allocation [81], among others.

2.3 Summary

This chapter provides a review of relevant literature is presented as a background for the goals of this dissertation. The key themes for this literature review are

1. The complexity, diversity, and flexibility of the WTA problem
2. The flexibility and applicability of approximate dynamic programming as a solution for resource allocation problems

Given the literature, there are gaps which must be covered to address the motivating problem. First, a more practical formulation for the DWTA must be formulated that considers dynamic weapons capabilities. Development of this new formulation requires solution methodologies not found in the literature. As a solution methodology, approximate dynamic programming is often used for large resource allocation problems. However, because of the structure and complexity of the WTA problem, the size of the decision space is often prohibitively large. Therefore, approximate dynamic programming methodologies which address this issue are investigated. This research developed in this dissertation specifically addresses each of these gaps.

III. Optimal multi-stage allocation of weapons to targets using adaptive dynamic programming

3.1 Abstract

We consider the optimal allocation of resources (weapons) to a collection of tasks (targets) with the objective of maximizing the reward for completing tasks (destroying targets). Tasks arrive in two stages, where the first stage tasks are known and the second stage task arrivals follow a random distribution. Given the distribution of these second stage task arrivals, simulation and mathematical programming are used within a dynamic programming framework to determine optimal allocation strategies. The special structure of the assignment problem is exploited to recursively update functional approximations representing future rewards using subgradient information. Through several theorems, optimality of the algorithm is proven for a two-stage Dynamic Weapon-Target Assignment Problem.

3.2 Introduction

The weapon-target assignment (WTA) problem is a model of combat operations where we maximize the total expected damage caused to the enemy's targets (or minimize the value of leaker missiles) using a finite number of weapons. Optimally assigning interceptors to targets is a subject that has become increasingly important with the proliferation of intercontinental ballistic missiles (ICBMs). The WTA problem is known to be NP-complete [60]. In general, two cases of the WTA problem are considered, *static* and *dynamic*. The static case allocates m weapons to n targets at one time after all problem information is known. The dynamic case provides an allocation policy over some time horizon, for which more information may arrive as time progresses. Generally, both formulations contain at least stochastic single shot

kill probabilities for weapon-target pairs, and many include additional uncertainties. One example of a dynamic problem is as follows. Suppose there are two waves of incoming ICBMs where the number of targets (ICBMs), n, and their values, V_j , in the first wave is known and the second wave is known only up to a probability distribution. If the single shot probability of the weapon (interceptor) successfully hitting a target is p, and each shot's outcome is independent of the outcome of any other shot, then the decision space for a fixed number of interceptors consists of how many interceptors to allocate to the first wave verses the number of inceptors allocated for assignment to the second wave. This formulation is attributed to Murphey [71] who proposes a stochastic decomposition approximation technique. This chapter provides an optimal solution for the formulation of [71] by exploiting the special structure of the problem.

3.3 Literature Review

3.3.1 Static Weapon-Target Assignment.

The SWTA is formulated as follows. Let V_j denote the value of the j^{th} target, W_i denote the number of available weapons of type i. We assume we have m weapon types and n targets. Let p_{ij} be the single shot probability that a weapon of type i will kill a target of type j, such that the single shot probability of survival is $q_{ij} = 1 - p_{ij}$. Our decision variable x_{ij} is the number of weapons of type i assigned to target j. The defensive SWTA problem is then formulated as a nonlinear integer program:

$$\min \sum_{j=1}^{n} V_j (\prod_{i=1}^{m} q_{ij}^{x_{ij}}) \tag{3.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \leq W_i \text{ for all } i = 1, 2, \ldots, m, \tag{3.2}$$

$$x_{ij} \geq 0 \text{ and integer, for all } i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \tag{3.3}$$

Much of the WTA literature has been dedicated to the SWTA problem formulation which was shown to be *NP-complete* in 1986 by Lloyd and Witsenhausen [60]. As such, much research has been done in the past several decades to determine efficient methods of identifying optimal solutions. Computationally efficient optimal methods exist for two cases of the SWTA under simplifying assumptions. First, given a homogeneous weapon set, $p_{ij} = p_j$ for all i, denBroeder [30] shows optimality is achieved by evenly distributing the weapons across as many targets as possible using the maximum marginal return (MMR) algorithm. The second instance assumes that each target can have at most one weapon assigned to it [24] [75]. Because our problem focuses on a special instance of the dynamic WTA (DWTA) problem, we focus our literature review there.

3.3.2 Dynamic Weapon-Target Assignment.

Though it has not been researched to the extent of the SWTA problem, the DWTA problem provides a more practical implementation by considering the impact current decisions have on future states. However, by breaking the problem up into several decision epochs, the DWTA is a much more complex problem. Similar to the SWTA, numerous methods have been employed to provide solutions for various types of DWTA problems. As the originator of the dynamic instance, Hosein [47] provides several results which are generalizable to the DWTA problem. Additionally, Castañon [20] and others at ALPHA TECH were developing advanced algorithms for the DWTA in parallel. Murphey [70] [71] uses stochastic decomposition for the

two-stage problem defined in Sect. 3.4. Chang [24] uses a static WTA approximation scheme within an iterative linear network flow framework to efficiently provide high-quality solutions for the DWTA. Because of the integer restriction for the decision variables, the chromosome representation within a GA presents a useful scheme for solving both the static and dynamic versions of the WTA problem. As such, much work has developed hybrid GAs to assist in solving the DWTA. Wu *et al.* [99] apply a modified GA to the DWTA and introduces weapon use deadlines within the problem formulation. Xin *et al.* [101] develop a heuristic which uses problem information (domain knowledge) and constraint programming to assign priorities to assignments. Evolutionary heuristics which use a hybridized GA with memetic algorithms have also been applied to the DWTA [25]. Additionally, Khosla [54] applies a hybrid heuristic which uses a simulated annealing (SA) type heuristic to determine the fitness of a population within a GA framework. Other heuristic techniques applied to the DWTA include Tabu Search [102], ACO with tabu table updates [103], and a modified Hungarian method with PSO [56] (though this is in an open source text, so it's rigor may be unverified). Lastly, exact dynamic programming [89] [91] has also been applied to the DWTA.

3.4 Problem Formulation

As a simplification to the multi-stage problem posed by Hosein [48], Murphey [69] defined a two-stage stochastic programming model of the DWTA problem. In this model, we consider the probability that an adversary has a total stockpile of weapons and shoots a portion of them in the first stage, with the remainder of the weapons known to a probability distribution. Let n_1 targets arrive in stage 1 with certainty, and n_2 targets arrive in stage 2 according to a known distribution. Let the random vector $\omega \in \Omega$ denote the number of second stage target arrivals where Ω is the set of

all possible arrivals. Suppose the probabilities of survival, q_j, and target values, V_j, for each target j are given. Then the 2-stage WTA programming formulation is:

$$Z_1(x) = \max_x \sum_{j=1}^{n_1} V_j^1 (1 - (q_j^1)^{x_j^{(1)}}) + \mathbb{E}_{\omega \in \Omega}[Z_2(x^{(2)}, \omega^j)] \tag{3.4}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n_1} x_j^{(1)} &\leq M, \\ x^{(1)} &\leq b \\ x_j^{(1)} &\in \mathbb{Z}^+, j = 1 \ldots, N \end{aligned}$$

$\mathbb{E}_{\omega \in \Omega}[Z_2(x^{(2)}, \omega^j)]$ is the expected second stage value and, given a number of second stage weapons, $x^{(2)}$, and a sample realization of targets, ω^j, is piecewise integer concave (for a proof of this, see [2]) and is solved using the MMR algorithm. $\sum_{j=1}^{n_1} x_j^{(1)} \leq M$ is the resource capacity constraint, and b is the vector denoting the maximum weapons that can be assigned to any one target.

$Z_2(x^{(2)}, \omega^j)$ is the solution to the second stage problem and is expressed as:

$$Z_2(x^{(2)}, \omega^j) = \max_{x^{(2)}} \sum_{j=1}^{n_2(\omega)} V_j^2(\omega)(1 - q_j^2(\omega)^{x_j^{(2)}}) \tag{3.5}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n_1} x_j^{(1)} + \sum_{j=1}^{n_2(\omega^j)} x_j^{(2)} &= M, \\ x^{(2)} &\leq b \\ x_j^{(2)} &\in \mathbb{Z}^+ \end{aligned}$$

3.5 Theoretical Results

In this section we discuss the methodology used to solve the above two-stage DWTA problem presented above and present the theoretical results for our solution. Instead of using the cutting plane approach of Murphey [70], we formulate the problem as a dynamic programming problem to develop a solution algorithm with the help of the post-decision dynamic programming formulation and the concave adaptive value estimation (CAVE) functional approximation algorithm developed by Godfrey and Powell [39].

3.5.1 Adaptive Dynamic Programming.

Consider a general finite space and discrete time horizon dynamic programming problem. Let $\mathcal{S}$ be the state space of the system with time horizon $t = 0, \ldots, T$. The state $S_t \in \mathcal{S}$ represents the state of the system at time t, and a decision x_t that acts on the system is selected from a finite set $\mathcal{U}$ at each time step. W_t is a random occurrence generated with a known probability distribution and the system evolves according to a transition function which has the form

$$S_{t+1} = f_1(S_t, x_t, W_t) \tag{3.6}$$

where $f_1(\cdot)$ is a function describing the system dynamics. Next, define the one-period contribution for being in state S_t and making decision x_t as $C_t(S_t, x_t)$ and express the T-stage value to be maximized as the expected value of the summation of the T costs:

$$\max_{x_t \in \mathcal{U}(S_t)} \mathbb{E}\left\{ \sum_{t=0}^{T} C_t(S_t, x_t) | S_0 \right\} \tag{3.7}$$

It is well known that problems of the form given in (3.7) can be solved by Bellman's optimality equations [12]:

$$J_t(S_t) = \max_{x_t}(C_t(S_t, x_t) + \mathbb{E}_{W_t}\{J_{t+1}(S_{t+1}(S_t, x_t, W_t))|S_t\} \tag{3.8}$$

Problems of this type grow exponentially within the state, decision, and outcome spaces - known as the curses of dimensionality. Therefore it is necessary to approximate the value function $J_{t+1}(S_{t+1}(\cdot))$. Adaptive Dynamic Programming provides a means for stepping forward through time iteratively using sample realizations of our approximated value function.

3.5.2 Two-Stage DWTA ADP Solution.

Our method uses Monte Carlo sampling of second-stage target arrivals to approximate our value function. By making use of the concavity of the stage 2 function, we have developed an algorithm which optimally determines the number of interceptors needed in the second stage. Given a fixed number of weapons and a sample realization of stage 2 targets, n_2, it is clear that $f(n_2) = \max_{x^{(2)}} \sum_{j=1}^{n_2} V_j(1 - q_j^{x_j^{(2)}})$ is a piecewise integer concave function. Here, $x_j^{(2)}$ denotes the number of weapons allocated to the j^{th} target in the second stage.

Using the post-decision state dynamic programming notation of Powell [83], if we assume a piecewise linear concave approximation, then our second stage post decision value function becomes $J_1^x(S_1^x) = \mathbb{E}_{\omega\in\Omega}[\hat{Z}_2(x^{(2)}, \omega^j)]$. Our post-decision state is then $S_1^x = n_2$, and for any given number of weapons, the slopes of our function represent the marginal value of adding one more weapon to the second stage. As such, we modify the MMR algorithm of denBrodeur [30] while maintaining optimality for the special case of the DWTA.

Algorithm: MMR Plus

Step 0: Given $\bar{J}_1^x(S_1^x)$,

Initialize $x_j = 0 \ \forall j = 1, \ldots, N$ and set $x_{N+1} = x^{(2)}$

Set $S_j = V_j$ for $j = 1, \ldots, N$.

Compute the marginal returns $MR_j = S_j(1 - q_j), MR_{N+1} = J_1^x(1) - J_1^x(0) \ \forall j$.

Initialize weapon index $i = 1$.

While $i \leq M$, *do*

Step 1: Find target k for which weapon i has the greatest effect Compute $k = \arg\max_{j=1,\ldots,N+1} MR_j$

Step 2: Increment the allocation to target k: $x_k \leftarrow x_k + 1$.

If $j \leq N$, update the expected surviving value $S_k = S_k q_k$, then update the marginal return $MR_k = S_k(1 - q_k)$,

else increment $x_{N+1} \leftarrow x_{N+1}+1$ and update the marginal return $MR_{N+1} = J_1^x(x_{N+1} + 1) - J_1^x(x_{N+1})$

set $i = i + 1$ and continue

We now prove the existence of a piecewise linear concave function and the optimality of the MMR Plus Algorithm.

Theorem 3.5.1 *If $J_1^x(S_1^x = n_2) = \mathbb{E}[Z(x^{(2)}, \omega)]$, the MMRPlus algorithm is optimal.*

Proof: Given any scenario in stage 2, by the MMR algorithm for the WTA problem, the solution is monotonic increasing and integer concave [30]. We represent the slopes of each stage 2 function for the n scenarios as

$$\begin{aligned} \xi_1^1 \geq \quad & \xi_2^1 \geq \xi_3^1 \geq \dots \\ & \vdots \\ \xi_1^n \geq \quad & \xi_2^n \geq \xi_3^n \geq \dots \end{aligned} \tag{3.9}$$

where ξ_i^j denotes the marginal reward gained by saving the i^{th} weapon for the second stage, given the j^{th} target arrival scenario. Let p_j, $j = 1, \dots, n$ be the probability of scenario j. Then, since $p_j > 0$, $\forall j$, the inequalities remain valid by multiplying the inequalities by their respective probabilities

$$\begin{aligned} p_1\xi_1^1 \geq \quad & p_1\xi_2^1 \geq p_1\xi_3^1 \geq \dots \\ & \vdots \\ p_n\xi_1^n \geq \quad & p_n\xi_2^n \geq p_n\xi_3^n \geq \dots \end{aligned} \tag{3.10}$$

Through term by term addition, the inequalities hold to obtain

$$\sum_{j=1}^{n} p_j\xi_1^j \geq \sum_{j=1}^{n} p_j\xi_2^j \geq \sum_{j=1}^{n} p_j\xi_3^j \geq \dots \tag{3.11}$$

Therefore, $\mathbb{E}[Z(x^{(2)}, \omega)]$ is monotone increasing and integer concave in $x^{(2)}$.

The resulting optimization problem at stage 1 is an integer optimization with a monotropic value function for which each separable function is monotonic increasing and piecewise integer concave. Since there is a single linear constraint coupling the allocations to all targets, strong duality guarantees the existence of a scalar dual variable λ such that, at the optimal allocations x_j^*, $j = 1, \dots, N+1$, the right derivatives

of each separable function are less than or equal to λ, and the left derivatives are greater than or equal to λ, and $\sum_{i=j}^{N+1} x_j^* = M$.

Since all the separable functions are piecewise integer concave, each function has a finite number of slopes (derivative values). The MMRPlus algorithm searches over the possible slopes of all the separable functions in decreasing order, modifying the allocations x_j appropriately until $\sum_{j=1}^{N+1} x_j = M$.

Since each function for each target and the function for the second stage expected value is monotonic increasing, by the property of the MMRPlus selecting the function with the greatest increase in objective value the MMRPlus algorithm is optimal. $\square$

So that we do not have to compute a piecewise linear approximation $\bar{J}_1^x(S_1^x)$ for every $x^{(1)}$ by simulation for every ω, our approach only focuses around the optimal value of $x^{(1)}$. To find the piecewise-linear approximation $\bar{J}_1^x(S_1^x)$ around the optimal $x^{(1)}$, we use a version of the CAVE algorithm developed by Godfrey and Powell while preserving concavity [37]. Given a realization of second stage target arrivals ω and the current solution of the MMRPlus algorithm where $x^{(2)} = M - x^{(1)} = x_{N+1}$ weapons are allocated to the second stage, the left and right derivatives, $v^-(\omega)$ and $v^+(\omega)$ respectively, are calculated as:

$$v^-(x^{(2)}, \omega) = \begin{cases} Z(x^{(2)}, \omega) - Z(x^{(2)} - 1, \omega), & \text{if } x^{(2)} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

$$v^+(x^{(2)}, \omega) = Z(x^{(2)} + 1, \omega) - Z(x^{(2)}, \omega) \tag{3.13}$$

where $Z(x^{(2)}, \omega)$ is the solution by the MMR algorithm of the second stage problem given $x^{(2)}$ weapons and sample realization ω.

Of course these are the left and right derivatives for only one sample realization of the problem and for only one particular state which is sufficiently captured in the number of weapons passed to the second stage,$x^{(2)}$. The piecewise linear approximation of $\mathbb{E}_\omega[Z(x^{(2)},\omega)]$ is defined by a finite set of ordered breakpoints, $\{(v^k,u^\cdot)|k\in\mathcal{K}\}$, where $\mathcal{K}=\{0,1,\ldots,M\}$. Each breakpoint defines a linear segment with v^k as the slope of the segment projected from u^k where a breakpoint is defined at each positive integer up to, and including, $M-1$. Concavity implies that the slopes are nonincreasing, as $v^0\geq v^1\geq\ldots\geq v^{M-1}$. By Theorem 3.5.1, $P(v^-(x^{(2)},\omega)\geq v^+(x^{(2)},\omega))=1$, $\forall x^{(2)}\geq 0$ since the slopes are always monotone decreasing and positive for all realizations of targets in stage 2. From the solution of the subproblems by the MMR algorithm, this can easily be proven for the 2-stage DWTA.

The left subgradient $v^-(x^{(2)},\omega)$ is smoothed into the approximation slopes to the left of $x^{(2)}$ to some minimal extent determined by the interval $I=[\max(0,\min(x^{(2)}-\epsilon^-,u^{k^-})),\min(\max(s+\epsilon^+,u^{k^++1}),M)]$. The same idea is applied to the right of $x^{(2)}$ for the right subgradient. Concavity is preserved by the following theorem, similar to the one found in [37]:

Theorem 3.5.2 *Consider a concave approximation defined by breakpoints* $\{(\nu^k,u^k)|k\in K\}$ *where* ν^k *are the integers in* $\{0,\ldots,M-1\}$. *Using the CAVE algorithm described above to obtain* $I=[u^m,u^n]$ *with post-decision state,* $x^{(2)}$. *Concavity is preserved under the smoothing operation where* $0<\alpha<1$.

Proof: Case I: If $u^m=u^n$ no update takes place and concavity of the original function is preserved.

Case II: $I\neq\emptyset$, then we use the updates

$$\nu^k_{new}=\alpha v^-(x^{(2)},\omega)+(1-\alpha)\nu^k_{old}\text{ for }k=m,\ldots,x^{(2)}-1 \tag{3.14}$$

and

$$\nu^k_{new} = \alpha v^+(x^{(2)}, \omega) + (1-\alpha)\nu^k_{old} \text{ for } k = x^{(2)}, \ldots, n-1 \tag{3.15}$$

The slopes of the original function decrease monotonically in k.

$$\nu^m \geq \cdots \geq \nu^{x^{(2)}} \geq \nu^{x^{(2)}+1} \geq \cdots \geq \nu^{n-1} \tag{3.16}$$

$$(1-\alpha)\nu^m \geq \cdots \geq (1-\alpha)\nu^{x^{(2)}} \geq (1-\alpha)\nu^{x^{(2)}+1} \geq \cdots \geq (1-\alpha)\nu^{n-1} \tag{3.17}$$

$$\begin{aligned} \alpha v^-(x^{(2)}, \omega) + (1-\alpha)\nu^m \geq \cdots \geq \alpha v^-(x^{(2)}, \omega) + (1-\alpha)\nu^{x^{(2)}} \geq \\ \alpha v^+(x^{(2)}, \omega) + (1-\alpha)\nu^{x^{(2)}+1} \geq \cdots \geq \alpha v^+(x^{(2)}, \omega) + (1-\alpha)\nu^{n-1} \end{aligned} \tag{3.18}$$

Equation 3.16 holds by the concavity of the original function. Equation 3.17 holds by multiplication of a positive constant. Equation 3.18 holds since $v^+(x^{(2)}, \omega) \leq v^-(x^{(2)}, \omega)$. Therefore, the resulting function is also concave. □

3.5.3 The Adaptive DWTA Algorithm.

Having explained the components of the algorithm, the MMR, MMRplus and CAVE algorithms are combined to form the solution algorithm. We let QApprox represent the current approximation and use the following algorithm to obtain our approximate dynamic programming solution:

Step 1 Initialization

- $j = 0$
- Set $\nu^i = 0, \forall i = 0, \ldots, M-1$
- Set $u^i = i, \forall i = 0, \ldots, M-1$
- $\epsilon^- = 2, \epsilon^+ = 2$

Step 2 Forward Simulation

- Solve current problem with current QApprox using MMRplus
- Generate second stage target random sample, $\omega \in \Omega$
- If $j > 20$ then $\epsilon^- = 1, \epsilon^+ = 1$
- $\alpha = 1/(1+j)$

Step 3 Value Function Update

- Determine the left and right derivative, $v^-(x^{(2)}, \omega)$ and $v^+(x^{(2)}, \omega)$, respectively using MMR.
- Update QApprox using the CAVE algorithm.
- If no change in 10 iterations STOP, else j=j+1 and return to Step 2.

We initially set $\epsilon^- = \epsilon^+ = 2$ to allow the update of the piecewise linear concave approximation of the value function to affect, at a minimum, an interval of size four. Each possible integer value for the approximation is not sampled infinitely often, so ϵ^- and ϵ^+ allows the stochastic subgradients to be averaged over a greater interval. The piecewise linear concave approximation of the value function is only repetitively updated in the neighborhood of the optimal integer value for the second stage decision. Therefore, the shape of the approximate value function for integer values far from the optimal integer value may be underestimates or overestimates of the true slope. However, because the function is concave, the critical region around the optimal integer value is the most sampled and accurate. The accuracy around the optimal integer value in the piecewise linear concave approximation of the value function is all that is needed to provide quality solutions.

After 20 updates we set $\epsilon^- = \epsilon^+ = 1$, allowing the minimum updates to occur only to the left and right slope of the piecewise linear concave approximation of the value

function. The size of the initial update interval and the rate at which the minimum interval is allowed to decrease is problem dependent.

Theorem 3.5.3 *Assume the optimal solution is (x_1, x_2), and that the subgradients D^+ and D^- for $\mathbb{E}[Z(x_2, \xi)]$ are known. Then, if $D^+(J_1^x(x_2)) = D^+$ and $D^-(J_1^x(x_2)) = D^-$, the MMRPlus algorithm will generate the optimal solution (x_1, x_2).*

Proof: As shown in Theorem 3.5.1 the function for $\mathbb{E}[Q(x_2, \xi)]$ is monotone increasing integer concave. Assume that the MMRPlus algorithm generates the solution $x_1, \ldots, x_{N+1}$ which is not optimal. Because the approximation is integer concave and increasing, if the stopping slope obtained by MMRPlus, λ is in the interval $[D^-, D^+]$, the MMRPlus algorithm will obtain an optimal solution, because all the marginal returns obtained for targets $j = 1, \ldots, N$ will be computed exactly. Hence, there are two possible cases where the optimal solution is not achieved:

Case I: $\lambda > D^+$. In this case, this requires that the MMRPlus algorithm find M slopes with values greater than $D+$. However, at the optimal solution, there are at most $M - n_2$ slopes associated with targets $j = 1, \ldots, N$ that are greater than $D+$. Since the approximation $\tilde{J}$ is integer concave, there are only $n_2 - 2$ slopes greater than D^+, the slope associated with the left derivative at n_2. This contradicts the statement that MMRPlus found M slopes with values greater than D^+.

Case II. $\lambda < D^-$. In this case, the MMRPlus algorithm found less than M slopes with values greater than or equal to $D-$. At the optimal solution, MMRPlus has $M - n_2$ slopes for $j = 1, \ldots, N$ that are greater than or equal to D^- which is greater than λ. Furthermore, the integer concavity of $\tilde{J}_2$ indicates that there are $n_2 + 1$ slopes greater than or equal to λ for $j = N + 1$. This contradicts the property that MMRPlus makes assignments in order of decreasing slopes, and stops after making M assignments. $\square$

Corollary 3.5.4 *The result in Theorem 3.5.3 can be relaxed to provide error bounds for convergence for* $D^+ - D^+(J_1^x(n_2))$ *and* $D^- - D^-(J_1^x(n_2))$*, so that the critical slopes only have to be accurate to a threshold* $\epsilon > 0$*.*

Proof: Assume that λ is the optimal solution of the SWTA problem using the full dynamic programming value-to-go function $J_1^x(n)$, and n_2 is the optimal allocation to stage 2. As long as the approximate value to go $\tilde{J}$ has the property that $\lambda \in [D^-(\tilde{J}_1^x(n_2), D^+(\tilde{J}_1^x(n_2)]$, the MMRPlus algorithm will find the optimal solution. Thus, $D^+(\tilde{J}_1^x(n_2)) \geq \lambda \geq D^-$ and $D^-(\tilde{J}_1^x(n_2)) \leq \lambda \leq D^+$.□

3.6 Computational Results and Conclusions

The Adaptive DWTA Algorithm works well for deterministic problems where all second stage targets arrive with probability 1. Then our algorithm is the equivalent of the MMR algorithm and yields the optimal solution. In this situation we are simply dividing a weapon target assignment problem between two stages and since the gradient of the first stage function is known to be piecewise linear concave [30], our piecewise linear concave function approximation is exact.

The first example problem has 8 targets in the first stage and up to 8 targets in the second stage, each with identical values, $V_j = 200$. There are 12 total weapons. We assume that the single shot probabilities of survival q_j are identical and set to 0.5. We assume that each of the 8 second stage targets has an actual probability of arrival of 0.5, where the arrival events of different targets are independent. Hence this leads to $2^8 = 256$ possible arrival events (scenarios) at the second stage. For this symmetric problem the optimal dynamic programming solution yields an optimal strategy that uses 8 weapons in the first stage and 4 weapons in the second stage. Then we conjecture that since the number of first stage targets is equal to the number of second stage targets with second stage targets having a probability of arrival of

0.5, then the number of weapons assigned to the first stage is twice that assigned to the second stage in the optimal allocation of weapons. Our algorithm obtains this result. The result of our algorithm is that 8 weapons are assigned to the first stage and 4 weapons are assigned to the second stage with a vector of slopes for the approximation function given as

$$(100.0, 100.0, 99.4, 80.4, 63.8, 63.8, 45.5, 0.0, 0.0, 0.0, 0.0, 0.0)$$

The marginal value of the first weapon assigned to a target in the first stage is 100, and the second weapon is 50. The 8 first stage targets all attribute a value of 100 to the objective value for the one weapon assigned to each target. The second stage approximate value function shows that the average net value of one more weapon, beyond the 4 already assigned, to the second stage is 63.8. Therefore, if an additional weapon became available it should be assigned to the second stage. The solution converged to the optimal assignment after 5 iterations which is significant less computation than explicitly determining the second stage required for using the 2^8 events to calculate the exact second stage function $\mathbb{E}[Q(n_2, \omega)]$.

The example was run again with 13 weapons for our second example. The allocation of the weapons was 8 to the first stage and 5 to the second stage as expected with a vector of slopes for the approximation function given as

$$(100.0, 97.9, 97.9, 93.7, 64.3, 53.2, 53.1, 34.8, 0.0, 0.0, 0.0, 0.0, 0.0)$$

The vector is one component larger for the 13th weapon. The values around the 5th component should be close to the previous vector since we would expect this subgradient to be sampled relatively often. The other components are not the same because they are not in the critical region and are sampled rarely. It is easily seen

that the last five components of the vector should not be 0.0 but are never sampled in the construction of the approximate value function.

We then computed the close form expression for the recourse function and compare our experimental results with the exact slopes. The first six analytical slopes are

$$(99.61, 98.05, 91.80, 80.47, 63.87, 52.83)$$

The critical slopes are the 4th and 5th slopes for the first example and the 5th and 6th slopes for the second example since this is the critical area of the approximate function where the majority of updates occur. After 5000 iterations of the algorithm the approximated gradients are within .07 of both critical slopes for the first example and .43 and .37 of the 5th and 6th slopes, respectively, for the second example. Fortunately, 5000 iterations are not required as seen by obtaining the answer in 6 iterations of our algorithm, the slopes must only be within a threshold value of the optimal slopes as shown in Corollary 3.5.4.

For our third example we look at the same problem as described in our first and second example except that the probability arrivals are a realization of a uniform distribution $U(0, 1)$ for each second stage target and 50 weapons are assigned. The probabilities used for this example are

$$0.480488, 0.888801, 0.275961, 0.840961, 0.768530, 0.719374, 0.825271, 0.123142$$

The analytic slopes are

$$\begin{aligned}(\quad & 99.99, 99.90, 99.02, 94.69, 82.38, 65.09, 50.76, 47.40, 41.08, 40.00, \\ & 31.30, 31.30, 21.79, 21.79, 20.10, 15.51, 13.87, 13.87, 10.75, 10.75, \dots)\end{aligned}$$

The slopes found by the Adaptive DWTA Algorithm are

$$\begin{aligned}(\quad & 100.0, 100.0, 100.0, 50.0, 50.0, 50.0, 29.16, 29.16, 29.16, 24.66, 24.66, \\ & 24.66, 23.88, 23.78, 23.78, 22.83, 22.81, 13.87, 10.71, 10.71, \dots)\end{aligned}$$

The optimal solution was $(x_1, x_2) = (32, 18)$ and was converged to after 8 iterations. Looking at the slope for $x_2 = 18$ we see that the Adaptive DWTA Algorithm converged to the true analytic slope after 5000 iterations of the algorithm taking .218 seconds to converge.

As a fourth and final example, we look at larger example of 100 first stage targets, 100 second stage targets whose values are each different. The Adaptive DWTA Algorithm required 1.312 seconds for 5000 iterations and converged to the optimal answer in 1886 iterations. This is compared to the $2^{100} = 1.26765 \times 10^{30}$ scenarios that exist for calculation of the analytical solution of $\mathbb{E}[Q(x^{(2)}, \omega)]$.

As demonstrated above, our initial results are favorable. Favorable results are not surprising since we know by Theorem 3.5.1 that $\mathbb{E}[Q(x^{(2)}, \omega)]$ is concave in $x^{(2)}$. Therefore a piecewise linear concave approximation should be very descriptive if each slope for each integer value is sampled infinitely often which is NOT the case since we limit the number of iterations to 5000 for each experiment and the slope is repetitively sampled at the critical value of $x^{(2)}$. We have shown, however, that the slopes of the approximation are very close to the slopes of $\mathbb{E}[Q(x^{(2)}, \omega)]$ around the optimal solution. The slopes obtained by the Adaptive DWTA Algorithm have been proven sufficient through corollary 3.5.4 and shown through experimentation to obtain the optimal solutions.

The computational savings obtained by using the Adaptive DWTA Algorithm are illustrated by the fact that for the fourth example the solution was obtained in 1.312

seconds rather than calculating the analytical solution using 1.26765×10^{30} scenarios. The Adaptive SWTA Algorithm is shown to be a fast optimal approach.

This chapter develops a solution algorithm for a two-stage dynamic weapon-target assignment problem and proves solution optimality. Future work will relax weapon homogeneity assumptions, and investigate the impact of cost constraints and defense system sensor capability on solution quality.

IV. Adaptive Dynamic Programming for a Two-Stage Dynamic Weapon-Target Assignment Problem

4.1 Abstract

This research investigates the optimal allocation of weapons to a collection of targets over a two-stage time horizon with battle damage assessment which is more widely known as the shoot-look-shoot problem. A single wave of targets arrives in stage one and resources are allocated with the intent of maximizing the value of destroyed targets. The result of the first stage allocations is realized, and the value of destroyed targets is determined. The remaining resources are allocated to any remaining targets, results realized, and the additional value of targets destroyed is determined. Though the shoot-look-shoot problem is more often approached as a queueing problem where targets continually arrive, this chapter provides a two stage stochastic formulation. This research investigates allocation of stage dependent resources to non-homogeneous targets. An adaptive dynamic programming algorithm is developed which provides high-quality solutions in a fraction of the time necessary to compute an optimal solution and is scalable to large problems. The special structure of the assignment problem is exploited and subgradient information is used to update a functional approximation of future rewards.

4.2 Introduction

The subject of this chapter is an effective method for generating high-quality solutions to a two-stage weapon target assignment problem. The objective is to maximize the total expected damage caused to the enemy's targets using a finite number of weapons. Optimally assigning interceptors to targets is a subject that has become more critical with the increase in the technological sophistication of adversaries, and

the potential proliferation of intercontinental ballistic missiles (ICBMs). The WTA problem is known to be NP-complete [60].

In general, two cases of the WTA problem are considered, *static* and *dynamic*. The static case allocates m weapons to n targets at one time given all problem information is known. The dynamic case provides an allocation policy over some time horizon, for which more information may arrive as time progresses. Generally, both formulations contain at least stochastic single shot kill probabilities for weapon-target pairs, and many include additional uncertainties.

One example of a dynamic problem is as follows. Suppose there are two waves of incoming targets where the number of targets , n, and their values, V_j for $j = 1, 2, \ldots, n$, in the first wave is known for $j = 1, 2, \ldots, n$ and the second wave is known only up to a probability distribution. If the single shot probability of the weapon (interceptor) successfully hitting a target is p and each shot's outcome is independent of the outcome of any other shot, then the decision space for a fixed number of interceptors consists of how many interceptors to allocate to the first wave verses the number of inceptors allocated for assignment to the second wave. This formulation is attributed to Murphey [71] who proposes a stochastic decomposition approximation technique. Ahner and Parson [4] provide an effective algorithm which provides optimal solutions for the problem discussed above given a fixed number of homogenous weapons. This research extends the work of Ahner and Parson [4] by incorporating second stage target dependency on the first stage outcomes. Additionally, weapon capabilities vary across stages. The remainder of the chapter is structured as follows. A review of literature is given in Section 4.3 followed by the formal statement of the problem in Section 4.4. Next, the proposed solution methodology is covered in Section 4.5, followed by numeric results in Section 4.6. Finally, concluding remarks and discussion of future research resides in Section 5.6.

4.3 Literature Review

The weapon-target assignment problem is a well studied problem with various sub-formulations. This section presents a review of relevant literature for three of these formulations, beginning with the static weapon-target assignment problem.

4.3.1 Static Weapon-Target Assignment.

Much of the literature has been dedicated to the SWTA problem formulation, and several papers have been developed since the 2006 survey by Cai *et al.* [42]. As with many NP-complete or other combinatorial optimization problems, the existing literature applies a wide variety of methods to effectively solve the problem. Ahuja *et al.* [5] present the most cited results from recent times and give a benchmark for solution quality through lower bounding (for the minimization problem) techniques. Their formulation uses integer linear programming and as a general integer network flow problem using a minimum cost flow to determine a new lower bound (if minimizing). The authors also provide a very large-scale neighborhood improvement heuristic algorithm which quickly solves moderately sized instances (up to 80 weapons and targets) optimally while providing high-quality solutions for larger problems (up to 200 weapons and targets). As previously discussed, the earliest optimal methods were presented by denBroeder [30] under a homogenous weapon set assumption. His method is generally known as the maximum marginal return (MMR) algorithm (when considering the maximization problem) and assigns weapons sequentially to the weapon with the highest remaining value until all weapons have been allocated. This greedy method is also a fast method for bounding of the solution when the homogeneous weapons assumption has been relaxed. Chang *et al.* [24], and Orlin [74] developed optimal methods under the assumption that each target can have no more

than one weapon assigned to it. These methods exploit the underlying network flow structure of the SWTA problem.

Since the first approximation technique for the SWTA was done in 1966 [28], a gamut of popular metaheuristics have been applied to the SWTA problem. This includes ant colony optimization (ACO) [57][88], particle swarm [34] [104] (of a slightly more generalized resource allocation problem), and genetic algorithms (GAs) [19] [58] [49] [61]. In addition, hybrid methods are used to provide solutions for the SWTA, to include ACO with SA [97], GA with ACO [33], GA using greedy eugenics to improve the quality of the offspring [59], and particle swarm with embedded greedy algorithms [50]. [95] provides a comparison of several heuristic algorithms for the WTA problem and poses a new hybrid algorithm consisting of particle swarm and random search to produce higher-quality solutions. In addition to these popular metaheuristic methods, several other approximation methods have been used for the SWTA. [26] uses a modified MMR type algorithm after changing the network representation from a one-to-many to a one-to-one mapping to efficiently approximate the optimal value. Rosenberger *et al.* [85] compares the sequential application of the auction algorithm in a greedy fashion to an exact (but computationally expensive) branching and bounding technique. [62] applies fuzzy reasoning to approximate optimum allocations in real-time for use on a battlefield. Lastly, Lagrangian relaxation [72] was used to decompose the problem into two tractable subproblems while iteratively updating the Lagrange multipliers. Though an extensive amount of research has been done into effectively providing high-quality solutions for the SWTA, none distinctly stand out as the best. Next, a more complex dynamic weapon target assignment formulation is presented, prior to providing a review of existing literature.

4.3.2 Dynamic Weapon-Target Assignment.

Though it has not been researched to the extent of the SWTA problem, the DWTA problem provides a more practical implementation by considering the impact current decisions have on future states. However, by breaking the problem up into several decision epochs, the DWTA is a much more complex problem. Similar to the SWTA, numerous methods have been employed to provide solutions for various types of DWTA problems. As the originator of the dynamic instance, Hosein [47] provides several results which are generalizable to the DWTA problem. Additionally, Castañon [20] and others at ALPHA TECH were developing advanced algorithms for the DWTA in parallel. Murphey [70] [71] uses stochastic decomposition for a slightly different two-stage problem. Chang [24] uses a static WTA approximation scheme within an iterative linear network flow framework to efficiently provide high-quality solutions for the DWTA. Because of the integer restriction for the decision variables, the chromosome representation within a GA presents a useful scheme for solving both the static and dynamic versions of the WTA problem. As such, much work has developed hybrid GAs to assist in solving the DWTA. Wu *et al.* [99] apply a modified GA to the DWTA and introduces weapon use deadlines within the problem formulation. Xin *et al.* [101] develop a heuristic which uses problem information (domain knowledge) and constraint programming to assign priorities to assignments. Evolutionary heuristics which use a hybridized GA with memetic algorithms have also been applied to the DWTA [25]. Additionally, Khosla [54] applies a hybrid heuristic which uses a simulated annealing (SA) type heuristic to determine the fitness of a population within a GA framework. Other heuristic techniques applied to the DWTA include Tabu Search [102], ACO with tabu table updates [103], and a modified Hungarian method with PSO [56] (though this is in an open source text, so it's rigor

may be unverified). Lastly, exact dynamic programming [89] [91] has also been applied to the DWTA.

4.3.3 Shoot-Look-Shoot.

The shoot-look-shoot class of problem is generally found in naval literature. Manor and Kress [64] provides optimality of a multi-stage greedy SLS solution assuming imperfect damage information. They also show that the original SLS problem is equivalent to a finite horizon deteriorating bandit problem, which dynamically allocates a single resource amongst a fixed number of arms. Aviv and Kress [8] evaluate several SLS tactics (such as the persistent shooter, fixed bound on munitions and dynamic bound on munitions) and analyzes their efficiency when damage information is uncertain (or incomplete). Glazebrook and Washburn [35] provide a brief survey of the SLS problem, and further investigate it by considering several scenarios in which information may be perfect or imperfect, the time horizon is finite or infinite, and homogeneity (or non-homogeneity) of weapons is considered. They approach the problem as a partially observable Markov decision process (POMDP), and apply dynamic programming citing the computational intractability of their methods as problem size increases. Yost and Washburn [105] also decompose the problem into a linear program to obtain an initial (bound) set of policies and dynamic programming to help improve the policies. The dynamic programming subproblem is also viewed as a POMDP, as in [35]. Karasakal [51] applies integer programming decomposition to determine SLS policies for allocating surface-to-air missiles within a naval task group. Castañon [23] approaches the SLS problem as a two stage resource allocation where the goal is to maximize the first stage allocations while considering the second stage recourse requirements. The formulation then takes on a similar form to that of the two-stage stochastic control problem defined by Murphey [71], and also looks

very much like a constrained two-stage form of Bellman's equation [10]. Linear interpolation and Lagrangian decomposition are then used to determine optimal recourse actions for the 2^{nd} stage. These values are then used recursively to greedily determine an approximate solution of the first stage problem.

4.4 Problem Formulation

Because it forms the basis from which the formulation is developed, the generalized static weapon target assignment (SWTA) is first introduced.

4.4.1 Static Weapon-Target Assignment.

The SWTA is formulated as follows. Let V_j denote the value of the j^{th} target, W_i denote the number of available weapons of type i. It is assumed that there are m weapon types and n targets. Let p_{ij} be the single shot probability that a weapon of type i will kill a target of type j, such that the single shot probability of survival is $q_{ij} = 1 - p_{ij}$. The decision variable x_{ij} is the number of weapons of type i assigned to target j. The SWTA problem is then formulated as a nonlinear integer program:

$$\min \sum_{j=1}^{n} V_j (\prod_{i=1}^{m} q_{ij}^{x_{ij}}) \tag{4.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \leq W_i \text{ for all } i = 1, 2, \ldots, m, \tag{4.2}$$

$$x_{ij} \geq 0 \text{ and integer, for all } i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \tag{4.3}$$

Much of the WTA literature has been dedicated to the SWTA problem formulation which was shown to be *NP-complete* in 1986 by Lloyd and Witsenhausen [60]. As such, a great deal of research has been done in the past several decades to determine

effective methods of identifying optimal solutions. Computationally efficient optimal methods exist for two cases of the SWTA under simplifying assumptions. First, given a homogeneous weapon set, $p_{ij} = p_j$ for all i, denBroeder [30] shows optimality is achieved by evenly distributing the weapons across as many targets as possible using the maximum marginal return (MMR) algorithm. The second instance assumes that each target can have at most one weapon assigned to it [24] [75]. Because this problem focuses on a special instance of the dynamic WTA (DWTA) problem, relevant literature from this class is reviewed.

4.4.2 Two-Stage Dynamic Weapon-Target Assignment.

Consider a problem where a single wave of targets arrives, and instead of allocating all weapons at once, it is done over two stages. Weapon capabilities are stage dependent, meaning that kill probabilities for stage one and stage two differ. Next assume the outcome of any stage one weapon allocations are determined prior to allocating additional resources in stage two. This problem is formulated as follows. Let N targets arrive in the first stage at which point x_{1j} shots are allocated to target j, for all $j = 1, 2, \ldots, N$. Let Ω be the set of all possible outcomes of the stage one allocations, then $\omega \in \Omega$ denotes a sample realization. This formulation allows for the single shot probabilities of survival, q_{tj}, $t = 1, 2$ to vary by stage, representing non homogeneity of weapons. This may be interpreted as two types of weapons, two weapon locations, or changing capabilities as time goes on. Let $\mathcal{J}_1$ be the set of targets for which weapons are allocated in stage one, target values, V_{tj}, $t = 1, 2$ transition probabilistically for each target $j \in \mathcal{J}_1$ from stage one to two. Let n_2 be the number of remaining targets after the stage one outcome has been determined. Next, x_{2j} weapons are allocated to each target $j \in \mathcal{J}_2(\omega)$ where $\mathcal{J}_2(\omega)$ is the set of targets in stage two, which depends on the outcome ω. Let C be the resource pool

from which weapons are selected. Additionally, there are m_1 weapons of type 1 and m_2 weapons of type 2. Then the 2-stage DWTA programming formulation is:

$$Z_1(x) = \max_x \sum_{j=1}^{N} V_{1j}(1 - (q_{1j})^{x_{1j}}) + \mathbb{E}_{\omega \in \Omega}[Z_2(x_2, \omega)] \tag{4.4}$$

where

$$Z_2(x_2, \omega) = \max_{x_2} \sum_{j=1}^{n_2} V_{2j}(\omega)(1 - (q_{2j})^{x_{2j}}) \tag{4.5}$$

subject to

$$x_1 + x_2 \leq C \tag{4.6}$$

$$x_1 \leq m_1 \tag{4.7}$$

$$x_2 \leq m_2 \tag{4.8}$$

$$x_{tj} \in \mathbb{N}; t = 1, 2; j = 1, \ldots, N; \tag{4.9}$$

$Z_2(x_2, \omega)$ is the expected second stage value and, given a number of second stage weapons, m_2, and a sample outcome of remaining targets, ω, is piecewise integer concave (for a proof of this, see [2]) and is solved using the MMR algorithm. Constraint (4.6) is the resource constraint, constraints (4.7) and (4.8) are the capacity constraints, and constraint (4.9) is the integrality constraint on the decision variables.

4.5 Methodology

Because of the special structure of this stochastic program, the work of [4] is extended to consider the case where a second stage approximation of the value function is used to allow for efficient solutions of the overall problem. This method uses Monte

Carlo sampling of the first stage outcomes to approximate the stage two value function. Because of the concavity of the stage two function the proposed algorithm is able to quickly determine an approximation of the tradeoff between using weapons in the first stage and using weapons in the second stage. Because the second stage utilizes a single resource class, the optimality of the MMR algorithm of denBroeder [30] is exploited to generate optimal allocations for any sampled $\omega \in \Omega$. This is further used to efficiently generate high quality solutions using my less computational time required to enumerate all possible first stage outcomes.

4.5.1 Adaptive Dynamic Programming.

Consider a general finite space and discrete time horizon dynamic programming problem. Let $\mathcal{S}$ be the state space of the system with time horizon $t = 0, \ldots, T$. The state $S_t \in \mathcal{S}$ represents the state of the system at time t, and a decision x_t that acts on the system is selected from a finite set $\mathcal{U}$ at each time step. W_t is a random occurrence generated with a known probability distribution and the system evolves according to a transition function which has the form

$$S_{t+1} = f_1(S_t, x_t, W_t) \tag{4.10}$$

where $f_1(\cdot)$ is a function describing the system dynamics. Next, define the one-period contribution for being in state S_t and making decision x_t as $C_t(S_t, x_t)$ and express the T-stage value to be maximized as the expected value of the summation of the T costs:

$$\max_{x_t \in \mathcal{U}(S_t)} \mathbb{E}\left\{ \sum_{t=0}^{T} C_t(S_t, x_t) | S_0 \right\} \tag{4.11}$$

It is well known that problems of the form given in (4.11) can be solved by Bellman's optimality equations [12]:

$$J_t(S_t) = \max_{x_t}(C_t(S_t, x_t) + \mathbb{E}_{W_t}\{J_{t+1}(S_{t+1}(S_t, x_t, W_t))|S_t\} \tag{4.12}$$

Problems of this type grow exponentially within the state, decision, and outcome spaces - known as the curses of dimensionality. Therefore it is necessary to approximate the value function $J_{t+1}(S_{t+1}(\cdot))$. Adaptive Dynamic Programming provides a means for stepping forward through time iteratively using sample realizations of the approximated value function.

4.5.2 Approximation of the Second Stage Value Function.

For any first stage weapon allocation, the number of outcomes grows exponentially in the number of weapons allocated and targets with weapons allocated to them. However, because of its concavity, estimates of the second stage value function are able to be generated by sampling outcomes of the first stage. The concave adaptive value estimation (CAVE) algorithm of Godfrey and Powell [38] provides such a method for approximation. CAVE uses stochastic subgradient information representing the marginal value for saving enough resources to use an additional weapon in stage two. The CAVE algorithm is shown in Algorithm 1.

4.5.3 Adaptive Dynamic Programming for a Two-Stage DWTA.

The CAVE algorithm is implemented within the MMRPlus algorithm of [4] to efficiently generate high quality solutions to the problem defined in (4.4) - (4.9). Powell's post-decision state notation [83] is adopted and the second stage post decision value function is defined as

Algorithm 1 Concave Adaptive Value Estimation (CAVE) Algorithm [38]

STEP 1 Initialization

- let $\mathcal{K} = \{0\}$, where $v^0 = 0, u^0 = 0$.
- set $\varepsilon^-, \varepsilon^+, \alpha$.

STEP 2 Collect Gradient Information

- Given a state $s \geq 0$, sample the gradients $\pi^-(s, \omega)$ and $\pi^+(s, \omega)$ with random outcome $\omega \in \Omega$

STEP 3 Define Smoothing Interval

- Let $k^- = \min\{k \in \mathcal{K} : v^k \leq (1-\alpha)v^{k+1} + \alpha\pi^-(s)\}$ and $k^+ = \max\{k \in \mathcal{K} : (1-\alpha)v^{k-1} + \alpha\pi^+(s)\} \leq v^k$
- Define the smoothing interval $\mathcal{Q} = \Big[\min\{s - \varepsilon^-, u^{k^-}\}, \max\{s + \varepsilon^+, u^{k^+}\}\Big)$.
- Create new breakpoints at s and the endpoints of $\mathcal{Q}$

STEP 4 Perform Smoothing

- For each segment in $\mathcal{Q}$, $v^k_{new} = \alpha\pi + (1-\alpha)v^k_{old}$ where $\pi = \pi^-(s)$ if $u^k < s$ and $\pi = \pi^+(s)$ otherwise.
- Adjust $\varepsilon^-, \varepsilon^+, \alpha$ according to step size rules.
- Return to Step 2.

$$J_1^x(S_1^x) = \mathbb{E}_{\omega\in\Omega}[Z_2(x_2,\omega)] \tag{4.13}$$

The post decision state is then $S_1^x = n_2$, and for any given number of weapons, the slopes produced by CAVE represent the marginal value of reserving a weapon for the second stage. The MMRPlus algorithm is shown in Algorithm 2.

Algorithm 2 MMRPlus Algorithm [4]

STEP 0 Initialization - Given $J_1^x(S_1^x)$

- $x_j = 0\ \forall j = 1,\ldots,N$ and set $x_{N+1} = x^{(2)}$
- Set $S_j = V_j$ for $j = 1,\ldots,N$.
- Compute the marginal returns $MR_j = S_j(1-q_j), MR_{N+1} = J_1^x(1) - J_1^x(0)$ $\forall j$.
- Initialize weapon index $i = 1$.

while $i \leq M$ **do**

- Find target k for which weapon i has the greatest effect, compute $k = \arg\max_{j=1,\ldots,N+1} MR_j$
- Increment the allocation to target k: $x_k \leftarrow x_k + 1$

if $j \leq N$ **then**

Update the expected surviving value $S_k = S_k q_k$, and update the marginal return $MR_k = S_k(1-q_k)$

else

Increment $x_{N+1} \leftarrow x_{N+1} + 1$ and update the marginal return $MR_{N+1} = J_1^x(x_{N+1}+1) - J_1^x(x_{N+1})$

end if

Set $i = i + 1$

end while

Using MMRPlus,the optimal allocation for stage one weapons is determined using the second stage approximation from the Monte Carlo experimentation. Given a realization of second stage target arrivals ω and the current solution of the MMRPlus algorithm where $x_2 = C - x_1$ weapons are allocated to the second stage, the left and right derivatives, $v^-(\omega)$ and $v^+(\omega)$ respectively, are calculated as:

$$v^-(x_2,\omega) = \begin{cases} Z(x_2,\omega) - Z(x_2-1,\omega), & \text{if } x_2 \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

$$v^+(x_2,\omega) = Z(x_2+1,\omega) - Z(x_2,\omega) \tag{4.15}$$

where $Z(x_2,\omega)$ is the solution by the MMR algorithm of the second stage problem given x_2 weapons and sample realization ω. This ensures that any excess resources after the first stage allocation are used in the second stage. The algorithm is presented as Algorithm 3. Note that the second stage probability function is dependent on the first stage allocation. Therefore, Equation (4.4) is not necessarily concave, and optimality is not guaranteed as in [4].

Algorithm 3 Adaptive Dynamic Programming Algorithm for 2 Stage DWTA

Initialize: $x_1 = m_1$, set $\varepsilon^-, \varepsilon^+, \alpha$,
Initialize: $v^- = v^+ = 0$, $\upsilon_i = 0$ for $i = 1, \ldots m_1$,
Set: $a = 1$, and fix *iterations*.
while $a < iterations$ **do**
  Determine optimal assignment of x_1 using MMR algorithm
  Using the assignment of x_1, generate outcome $\omega \in \Omega$ using Monte Carlo sampling
  Set $x_2 = C - x_1$
  if $n_2 > 0$ **then**,
    Determine optimal assignment of x_2 using MMRPlus - call it V
    Determine optimal assignment for $x_2 - 1$ and $x_2 + 1$ using MMRPlus - call them J^- and J^+, respectively
    $v^- = J - J^-, v^+ = J^+ - J$
  else
    $v^- = v^+ = 0$
  end if
  Update υ_i for $i = 1, \ldots m_1$ using CAVE algorithm
  $a = a + 1$, update $\varepsilon^-, \varepsilon^+, \alpha$
end while

4.6 Numeric Results and Discussion

Initially small scale experiments are run so that exact solutions can be determined for comparison of the proposed method. Two approximations are used as benchmarks for further comparison. These benchmark approximations are used because because of their computational simplicity as well as their ability to provide quality solutions for comparison. All experiments discussed herein use Matlab 2013a on a 3.07 GHz Intel Xeon with 24 GB RAM.

4.6.1 Small scale experiments.

To test the algorithm, 100 problem instances are randomly generated as follows. Integer target values are randomly generated, ranging from one to ten, and N is fixed. Survival probabilities are independently and randomly selected with $q_{tj} \sim UNIF(0.1, 0.4)$ for $t = 1, 2$ and $j = 1, 2, \ldots, N$. For the initial set of experiments, M is fixed at seven, N is varied at seven and eleven and the values computed. Two approximation schemes are selected for comparison of the proposed method. First, a greedy approximation developed by Castanon and Wohletz [22] is used that proves very effective for small scale tests, but suffers extensive computation time for larger problems. This algorithm is presented as Algorithm 4.

In order to describe the greedy approximation of [22], several items must be defined. Define $\mathbf{x}_1 = (x_{11}, x_{12}, \ldots, x_{1N})$ and let $\mathbf{x}_{1j}^{+} = (x_{11} \ldots x_{1j}\ x_{1j} + 1\ x_{1(j+1)} \ldots x_{1N})$. Let $\Omega = \{0, 1\}^N$ denote the outcome space for a given allocation where $\omega_j = 0$ denotes that target j has been destroyed, and $\omega_j = 1$ denotes target j survives. Given a stage one allocation $\mathbf{x}_1$, then

$$P(\omega|\mathbf{x}_1) = \prod_{\{j|\omega_j=0\}} (1 - (1 - p_{1j})^{x_{1j}}) * \prod_{\{j|\omega_j=1\}} (1 - p_{1j})^{x_{1j}} \tag{4.16}$$

In addition, define $\mathbf{V}(\omega) = \sum_{j|\omega_j=0} V_{1j}$, and

$$J(\mathbf{x}_1) = \sum_{\omega\in\Omega} \mathbf{V}(\omega)P(\omega|\mathbf{x}_1)J_2^*(\omega, M - \sum_{i=1}^{N} x_{1i}) \tag{4.17}$$

The greedy algorithm is then

Algorithm 4 2-stage greedy WTA algorithm [22]

Initialize: $x_{1j} = 0$, for $i = 1, 2, \ldots N$.
while $\sum_{j=1}^{N} x_{1j} < M$ **do**
  For each j, compute $MR_j(\mathbf{x}_1) = J(\mathbf{x}_1) - J(\mathbf{x}_{1j}^+)$
  Select j^* for which $MR_{j^*}(\mathbf{x}_1) \leq MR_j(\mathbf{x}_1^+)$ for all $j \neq j^*$.
  if $MR_{j^*} < 0$ **then**
    Set $x_{j^*} = x_{j^*} + 1$
  else *Break*
  end if
end while

Because of the second stage dependency on the first stage outcome, Algorithm 4 is not optimal. However, it has been shown to provide optimal solutions to randomly generated problems of smaller size [22], and is used to provide a metric for larger sized problems due to its relative computational tractability. In the second approximation, denoted MMR in the results tables, all possible combinations of x_1 and x_2 are generated. MMR is then run for each stage on every possible $x = (x_1, x_2)$, and the x which maximizes the sum of the two stage expected value is selected. Since it considers all possible outcomes, the exact expected target destruction value is reported for the CW heuristic. Because of the dependence on first stage outcomes, 1000 monte carlo simulations are run to determine the expected value for the MMR and ADP policies. The expected value for the MMR method is also presented for further validation Where appropriate, common random numbers were used to reduce experimental variation. The average optimality gap and associated standard deviations for the 100 experiments, are presented in Table 1. Additionally, computation time for

each method is reported in Table 2. The results of the first 10 experiments for each problem size are shown in Figure 2.

Table 1. Optimality gap (%) for 100 randomly generated problem instances

M	N	CW Heur. %Diff	ADP %Diff	MMR Sim % Diff
5	5	0.0	0.087 ±1.6	6.44 ±4.55
5	10	0.0	0.074 ±3.7	1.56 ±2.24
10	5	0.0	1.07 ±0.093	2.76 ±1.86
10	10	0.0	1.13 ±1.21	7.36 ±4.53

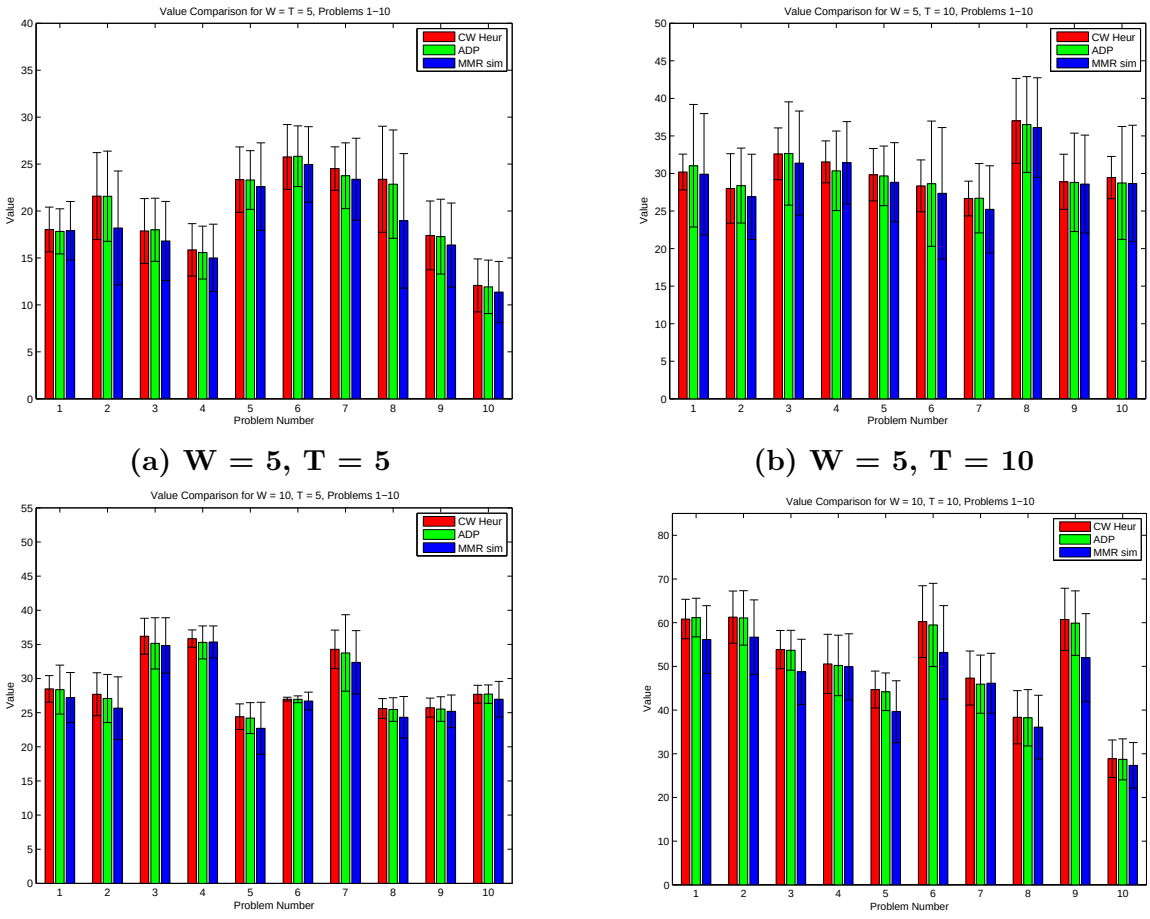
Table 2. Computation time (seconds) for 100 randomly generated problem instances

M	N	CW Heur	ADP	MMR
5	5	0.0058 ±0.0038	0.0815 ±.0214	0.0047 ±0.0112
5	10	0.0116 ±0.0064	0.0740 ±.0086	0.0034 ±0.0005
10	5	0.0192 ±0.0045	0.0779 ±0.0117	0.0072 ±0.0003
10	10	0.1020±0.0586	0.1044 ±0.0072	0.0073 ±0.0009

For this set of small scale experiments, solving exactly or using the first approximation methods is preferable. However, the value obtained through the ADP algorithm is very competitive, and the strength of the ADP method comes as problem size increases.

The next set of experiments varies the number of weapons and targets between 10 and 20 to determine the effectiveness of the method on slightly larger problem sizes. The CW heuristic and the two-stage MMR approximation remain the principal benchmarks. For these experiments, 50 test problems are randomly generated using the same parameters as above with 1000 simulations run on the solutions. Since the simulated MMR results provide a sufficient estimate of the approximation, they are reported for this analysis. The average percent difference from the CW heuristic for the ADP and MMR methods, are presented in Table 3, with computation times in Table 5.

The CW Heuristic was computed in a reasonable amount of time for problems with less than 40 weapons or targets. However, these problems take several minutes



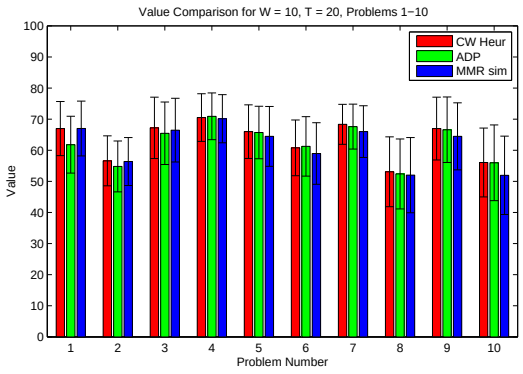
(a) W = 5, T = 5 **(b) W = 5, T = 10**

(c) W = 10, T = 5 **(d) W = 10, T = 10**

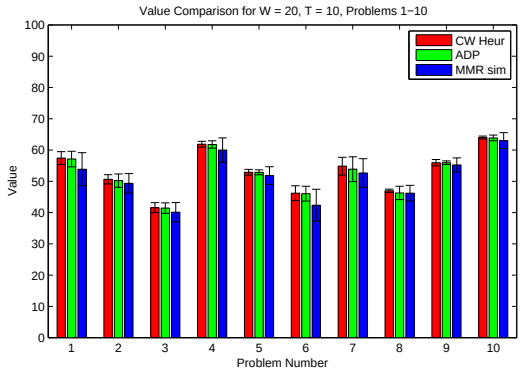
Figure 2. Results for first 10 small sized experiments at varying W & T

Table 3. Gap from CW Heur for 50 randomly generated medium sized problems

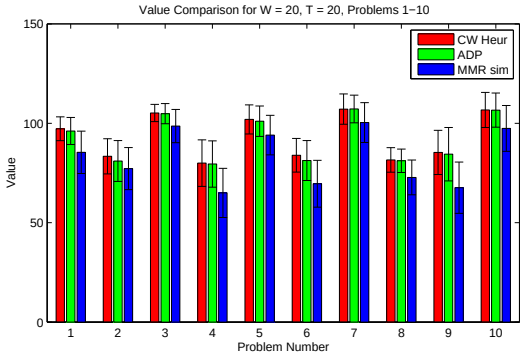
		Gap(%) ± (std dev)	
Weapons	Targets	ADP	MMR
10	20	0.55 ± 2.91	2.02 ± 2.09
20	10	0.8 ± 0.89	3.12 ± 2.07
20	20	0.87 ± 0.97	9.78 ± 4.17



(a) W = 10, T = 20



(b) W = 20, T = 10



(c) W = 20, T = 20

Figure 3. Results for first 10 medium sized experiments at varying W & T

each to solve, and in some cases, storage is a constraining factor. Hence, the percent improvement of the ADP method over that of the MMR method is used as the primary metric. The results of this analysis are shown in Table 4.

Table 4. Percent difference of ADP over MMR for 50 randomly generated medium sized problems

Weapons	Targets	% Δ (ADP - MMR)
10	40	-0.5 ± 2.2
20	40	0.39 ± 3.47
40	10	-0.12 ± 0.52
40	20	2.63 ± 2.66
40	40	8.4 ± 4.9

Table 5. Computational results for 100 randomly generated medium sized problems

		Comp Time (s) ($\pm$ std dev)		
Weapons	Targets	CW Heur	ADP	MMR
10	20	62.9445 ± 7.9483	0.1255 ± 0.0213	0.0082 ± 0.0008
10	40	-	0.1153 ± 0.0279	0.0073 ± 0.0005
20	10	54.4406 ± 9.4137	0.1529 ± 0.017	0.0233 ± 0.0017
20	20	93.8378 ± 13.7356	0.1938 ± 0.0152	0.0245 ± 0.002
20	40	-	0.197 ± 0.0282	0.0244 ± 0.0026
40	10	-	0.1906 ± 0.0125	0.0715 ± 0.0079
40	20	-	0.246 ± 0.0218	0.00714 ± 0.0032
40	40	-	0.3377 ± 0.0247	0.0861 ± 0.0043

Results show a statistically insignificant difference between the ADP method and MMR when the number of weapons is far less than the number of targets. This is an intuitive result because with few weapons, it will be optimal to spread them out as evenly as possible over the highest valued targets. This suggests that any approximation which reinforces this principle will generate very similar solutions. However, as the number of weapons increases to a level greater than or equal to the number of targets, the ADP outperforms on average. As further evidence of this, confidence intervals around the difference in the means between the 1000 monte carlo simulations were developed. Figures 4 and 5 demonstrate the significant improvement gained through the use of the ADP method as problem size increases, when there are more weapons than targets.

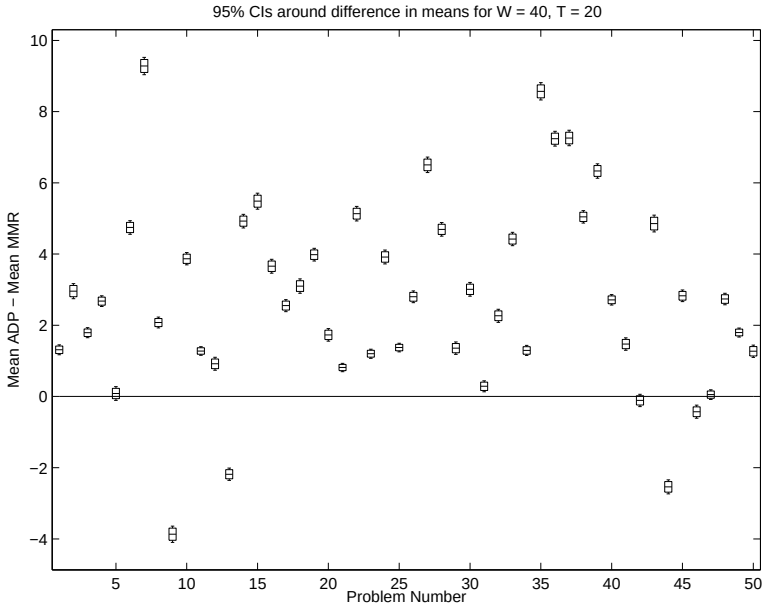


Figure 4. 95% CI's around difference in means ($\bar{X}_{ADP} - \bar{X}_{MMR}$)

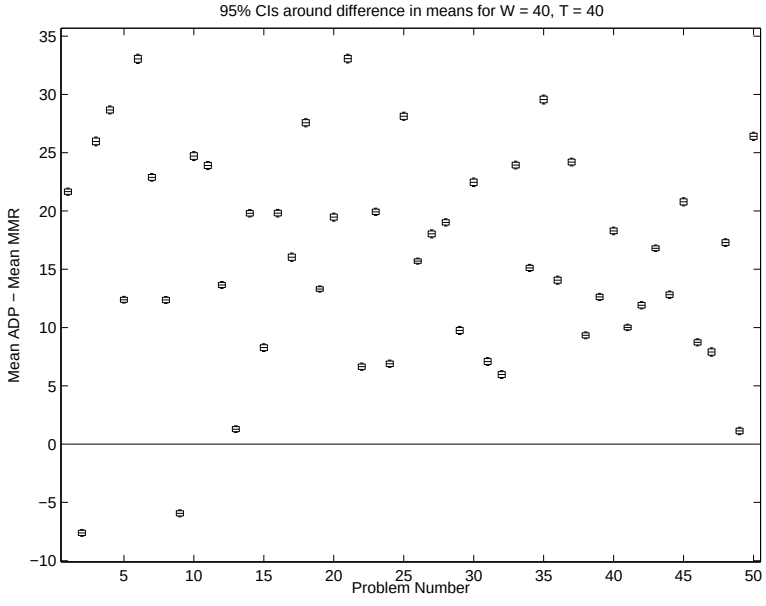


Figure 5. 95% CI's around difference in means ($\bar{X}_{ADP} - \bar{X}_{MMR}$)

Figure 4 shows approximately 90% of the randomly generated problem instances showing a statistically significant increase in value, while Figure 5 shows improvement 96% of the time. Again, because both methods spread weapons across the highest expected return, it is unsurprising that when the targets outnumber the weapons, both methods are fairly consistent. Conversely, when the number of weapons is relatively greater than the number of targets, the ADP substantially improves because it is accounting for the future value gained while generating first stage allocations. The MMR approximation generally pulls weapons to one stage or the other and allocates them fully, but the ADP method tends to spread weapons across stages, improving the likelihood that leakers will be destroyed in stage two. This is evidenced in the improvement for cases where the number of weapons and targets are equal. Computation times for the two methods are both less than a second, with the MMR method generally running faster, but with comparatively poorer performance. The next set of experiments are done to see how the methods perform on problems of a much larger size.

4.6.2 Large Scale Experiments.

For the large scale experiments, 50 randomly generated problem instances were run using the same parameters as in Section 4.6.1. For this set of experiments, the number of weapons and targets vary between 100, 200 and 400. Because of the insignificant improvement when there are more targets than weapons, this analysis focuses on the cases where the number of weapons are greater or equal to that of the targets. With problems of this size, solution of the greedy algorithm becomes intractable due to the potential size of the outcome space as the algorithm progresses. Therefore, the MMR becomes the sole benchmark to determine solution quality. Simulations are run on the policies of each method, and the results are presented in Table 6.

Table 6. Numerical results for 100 randomly generated large sized problems

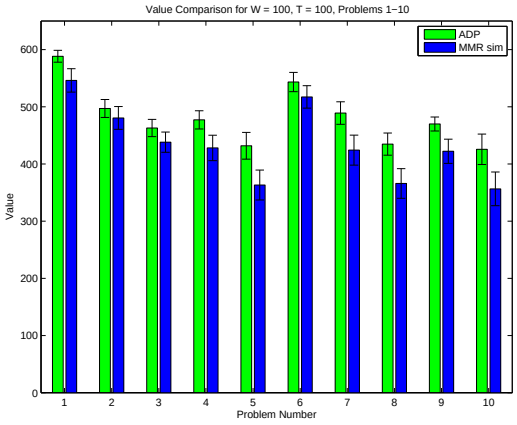
Weapons	Targets	% Difference of ADP vs. MMR
100	100	$9.19 \pm 4.12\%$
200	100	$3.68 \pm 2.75\%$
200	200	$9.06 \pm 5.11\%$
400	100	$0.22 \pm 0.37\%$
400	200	$4.04 \pm 2.57\%$
400	400	$9.65 \pm 4.32\%$

Table 7. Computation time (seconds) for 100 randomly generated large sized problems

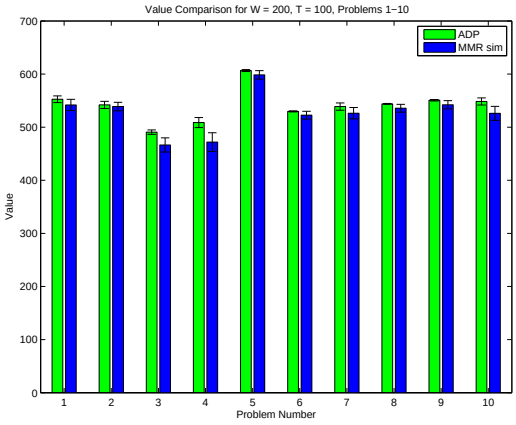
Weapons	Targets	ADP	MMR
100	100	1.2635 ± 0.0794	1.983 ± 0.067
200	100	1.3027 ± 0.893	2.0743 ± 0.0711
200	200	1.741 ± 0.203	2.3069 ± 0.2406
400	100	1.7233 ± 0.1198	6.2772 ± 0.1323
400	200	2.9153 ± 0.3817	9.5193 ± 1.1262
400	400	3.7753 ± 0.3781	10.4585 ± 0.5697

Results are consistent with the findings of Section 4.6.1, with the notable increase in performance of the ADP method. As problem size increases, the ADP method continues outperforming the two stage MMR. Additionally, computation time for the proposed method is much more competitive as problem size increases. For problems where there are many targets coming in at a time, this provides a quick approximation for determining the number of weapons to save for a second stage. Figures 6 and 7 present the simulated values and confidence intervals around the difference in the simulated means for the first ten problems of each large scale case.

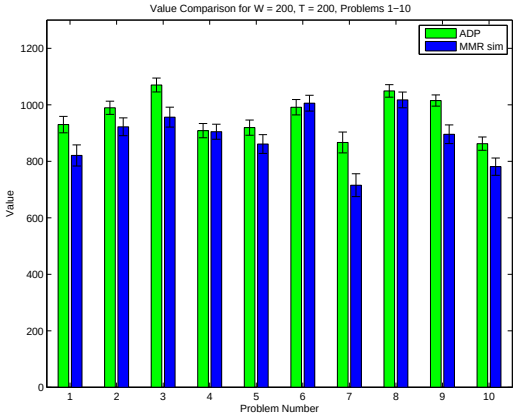
The black lines in Figures 7a-7f are at $y = 0$. Since the confidence intervals consistently above this line means that the null hypothesis that the difference in the means is zero is rejected and there is a significant difference. This is generally true in all cases except where there are 400 weapons and 100 targets. This is likely due to the large proportion of weapons to targets and the defined kill probabilities. The ADP method rarely under performs comparatively, and even when it does, the difference in destroyed target value is very small practically speaking. The speed of the ADP



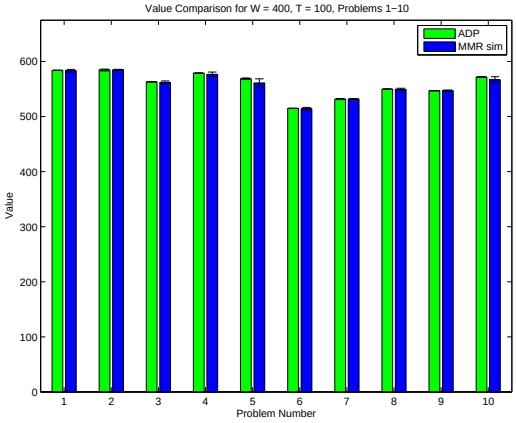
(a) W = 100, T = 100



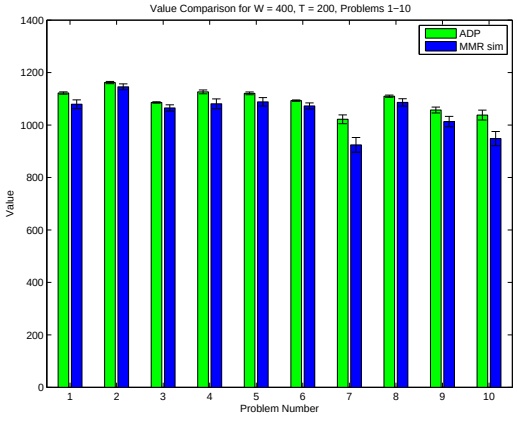
(b) W = 200, T = 100



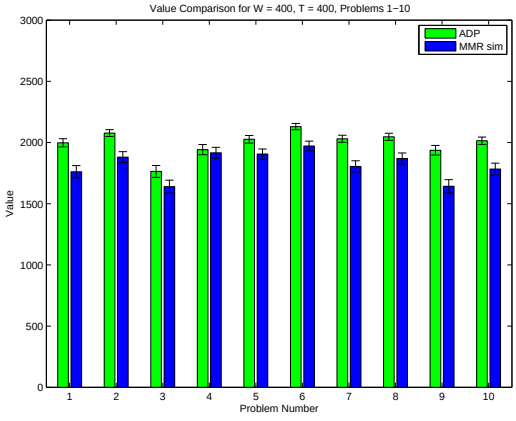
(c) W = 200, T = 200



(d) W = 400, T = 100

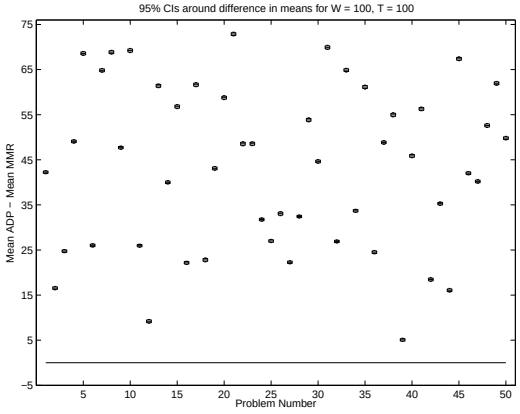


(e) W = 400, T = 200

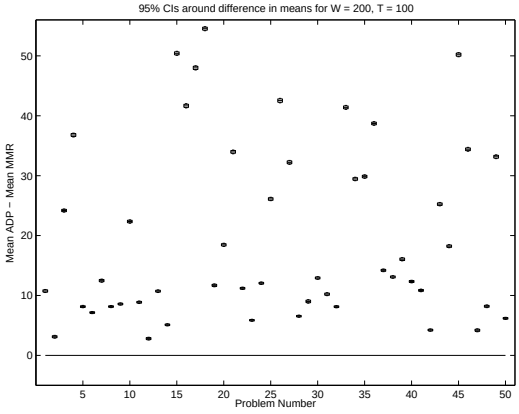


(f) W = 400, T = 400

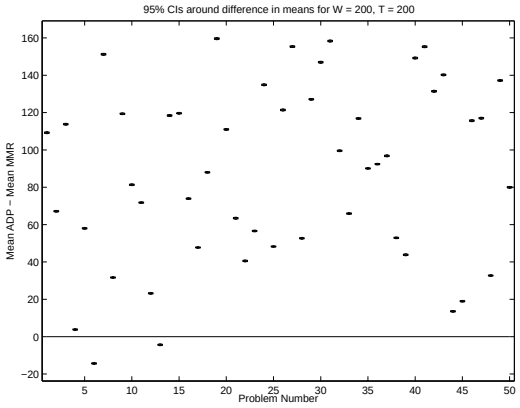
Figure 6. Results for first 10 large scale experiments at varying W & T



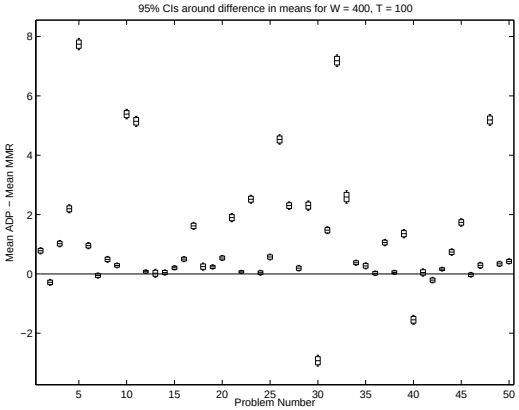
(a) W = 100, T = 100



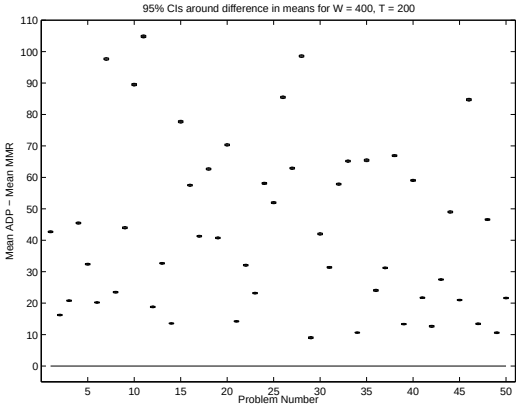
(b) W = 200, T = 100



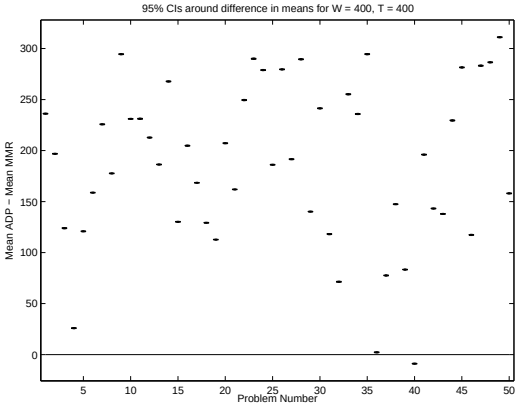
(c) W = 200, T = 200



(d) W = 400, T = 100



(e) W = 400, T = 200



(f) W = 400, T = 400

Figure 7. 95% CI's around difference in means ($\bar{X}_{ADP} - \bar{X}_{MMR}$)

algorithm, however, is nearly four times as long for the MMR method on the test problems, suggesting the desirability of the ADP method.

4.7 Conclusions and Future Research

This research develops an efficient solution algorithm for a two-stage shoot-look-shoot scenario where the second stage target set is dependent on the first stage allocations. Through Monte Carlo simulation, subgradients of the second stage value function are approximated. These subgradients are then used to get an approximation of the two-stage value for all first stage allocations. This method has been shown to be competitive with established techniques for small to medium sized problems, but preferred as problem size increases. The CW heuristic is able to address problem instances up to 20 weapons and 20 targets. For those problems, the proposed method obtained values within 1.2% of optimal solutions found using the CW heuristic. For large problems, the ADP approach consistently outperformed the MMR heuristic by up to 8.4% for small problems and up to 9.6% for larger problem instances. The ADP approach in [4] and further developed here offers significant flexibility to be extended to numerous other problem formulations. First, the algorithm can be extended to include the impact of cost on the approximation scheme as well as the effect sensors may have in the first stage, second stage, or both. Additionally, weapons for this effort are homogeneous within a stage, so a natural extension will investigate non-homogeneous weapons in and across stages. Last, because the subgradients represent the marginal increase in reserving a weapon for future stages, the algorithm may be very effective in instances where there are more than two stages. Therefore, additional research may extend this to multiple stages.

V. Approximate Dynamic Programming Methods for a Cooperative Dynamic Weapon-Target Assignment Problem

5.1 Abstract

The dynamic weapon target assignment (DWTA) problem is an extension of the static weapon-target assignment problem in which assignments are made over time, instead of all at once. This research investigates a cooperative version of the DWTA problem where the single-shot probability of kill is conditional upon the current target set. The sequential nature of the problem lends itself to solution by dynamic programming. However, because of the curses of dimensionality, large problems often become computationally intractable. An approximation method is proposed which reduces the size of the decision space to be investigated. Through ordinal optimization a rigorous method for ensuring selection of high-quality decisions from the action space for any given state is demonstrated. Various distributions are investigated, and results show that near optimal solutions can be obtained in much less computation time.

5.2 Introduction

The weapon-target assignment (WTA) problem is a fundamental resource allocation problem in the field of military operations research where the goal is to assign weapons to targets such that some objective is optimized based on the number of targets destroyed. Because of its applicability to numerous issues facing military analysts, such as ballistic missile defense, air-to-ground operations, and integrated air defense systems (IADS), this problem continues to be of significant operational importance. Additionally, because of the complexity of the various formulations, the WTA problem also maintains significance in the theoretic realm. Though it is also

found under other names, two general types of WTA problem exist: *static* (SWTA) and *dynamic* (DWTA). In both forms, there is a single-shot probability of kill for a given weapon-target assignment.

First proposed by Manne [63], the static formulation allocates a set of weapons to targets at one time given all problem information is known. Many variations of the SWTA exist within the literature (see [66] and [31]). The dynamic case provides an allocation policy over some time horizon, for which more information may arrive as time progresses.Many formulations of the DWTA are found, but for each, their underlying structure consists of the sequential allocation of weapons to targets with some sort of observed outcome occurring between stages. First formulated by Hosein [48], the dynamic problem has similar probabilistic characteristics as the static problem, but the complexity is increased with the inclusion of a solution over some time horizon. In the DWTA problem, weapons allocations impact the future state space. As such, the DWTA maintains increased complexity for which few solution techniques exist.

Though it has not been researched to the extent of the SWTA problem, the DWTA problem provides a more realistic implementation by including a temporal component. As such, the DWTA is a much more complex problem from a mathematical standpoint and has received a fair amount of attention in the literature. Similar to the SWTA, a number of methods have been employed to provide solutions for various types of DWTA problems. As the originator of the dynamic instance, [47] provides several results which are generalizable to the DWTA problem. Murphey [70] [71] use stochastic decomposition for a two-stage DWTA problem. Specific to the general DWTA problem, Chang *et al.* [24] use a static WTA approximation scheme within an iterative linear network flow framework to effectively provide high-quality solutions for the DWTA. Because of the integer restriction for the decision variables,

the chromosome representation within a GA presents a useful scheme for solving both the static and dynamic versions of the WTA problem. As such, much work has developed hybrid genetic algorithms (GAs) to assist in solving the DWTA. Wu *et al.* [99] apply a modified GA to the DWTA and introduce weapon use deadlines within the problem formulation. These deadlines follow the principles of scheduling theory, and are in the form of additional constraints such that a weapon must be shot at a target by a specified time or it is rendered unusable. The authors call their method a modified GA because it applies a basic GA iteratively, assigning a weapon to a target (possibly suboptimally) immediately before the deadline is reached. [101] develop a heuristic which uses problem information (domain knowledge) and constraint programming to assign priorities to assignments. Evolutionary heuristics which use a hybridized GA with memetic algorithms have also been applied to the DWTA by [25]. Additionally, [54] applies a hybrid heuristic which uses a simulated annealing (SA) type heuristic to determine the fitness of a population within a GA framework. Other heuristic techniques applied to the DWTA include Tabu Search by Xin *et al.* [102], ant colony optimization (ACO) with tabu table updates by [103], and a modified Hungarian method with particle swarm optimization (PSO) by [56], although this is provided in an open source text, so it's rigor may be unverified. Lastly, exact dynamic programming is applied to specific instances of the DWTA problem [91][89]

The rest of this chapter is structured as follows. Section 5.3 defines the problem and provides the modeling framework. Next, Section 5.4 presents the proposed methodology along with a presentation of some numeric examples and computational results in Section 5.5. Finally, conclusions and future research are presented in Section 5.6.

5.3 Problem Definition

This section provides the description of the problem, to include assumptions for the DWTA problem given in Section 5.3.1, followed by its formulation as an infinite horizon discrete time Markov decision process (MDP) in Section 5.3.2.

5.3.1 Problem Description.

Consider an offensive weapon target assignment problem consisting of a known set of weapons used to penetrate an integrated air defense system, of which all targets are assumed known. The DWTA divides the total duration of the attack into several discrete intervals in which information is obtained about the outcomes of the previous allocation. Any targets destroyed are not targeted in subsequent stages, allowing the operators to make better use of their weapons. To model the various layers of an IADS, the problem considers single-shot probabilities which depend on the current target set. The basic assumptions for this DWTA formulation are as follows:

- In each stage, a subset of the remaining weapons is selected and committed simultaneously
- The problem is solved at each stage using previous stage information
- Targets are present for the entire time horizon with an associated value; their value goes to zero when destroyed
- The outcome of each stage is perfectly observed prior to the next stage
- Computing the optimal assignment for the current stage always assumes optimal assignments will be made in subsequent stages
- Weapons are allocated at each stage with the goal of optimizing the objective value at the end of the final stage

- Geospatial characteristics of the weapons or targets are implicitly accounted for in their effect on the probability space

The elements of this multi-stage problem are next defined and the dynamic programming formulation provided.

5.3.2 Problem Formulation.

This problem is modeled as an infinite horizon, discrete time Markov decision process (MDP) using the collection of objects

$$\{\mathcal{T}, \mathcal{S}, \mathcal{A}, p(\cdot|S,a), C(S,a,W)\} \tag{5.1}$$

where $\mathcal{T}$ is the set of decision epochs, $\mathcal{S}$ is the state space, $\mathcal{A}_S$ represents the set of allowable actions given the system is in state S, with $\mathcal{A} = \bigcup_{S\in\mathcal{S}} \mathcal{A}_S$, $p(\cdot|S,a)$ is the probability transition function conditioned on being in state S and making decision $a \in \mathcal{A}_S$, and $C(S,a,W)$ is the reward obtained from being in state S, making decision a, and realizing the outcome W. Each of these elements are described in greater detail below.

Let $\mathcal{T} = \{1, 2, \ldots\}$ be the set of time stages and let $t \in \mathcal{T}$ denote a specific stage. Let $S_t = (R_t, Y_t) \in \mathcal{S}$ denote the state of the system at time t, where R_t is a vector indicating the number of weapons (of M different types) remaining in inventory and Y_t is a vector indicating the number of targets (of N different types) still functioning. $R_t = (R_{t1}, R_{t2}, \ldots R_{tM})$, where R_{tr} is the number of weapons of type r at time t, $r = 1, \ldots, M$. $Y_t = (Y_{t1}, Y_{t2}, \ldots Y_{tN})$, where Y_{ty} is the number of targets of type y at time t, each with associated value, V_y, $y = 1, \ldots, N$. A state $S \in \mathcal{S}$ corresponds to a particular pair of vectors indicating the number of weapons and targets remaining. Define $p_{ry|Y_t}$ as the single-shot probability of kill if weapon type r is allocated to target type y given the current target set Y_t. Define $q_{ry|Y_t} = 1 - p_{ry|Y_t}$ as the corresponding

probability of survival. The conditional probabilities of survival are used to model the cooperative nature of an IADS; as certain targets are destroyed, the attacker achieves improved probability of destroying other targets. For brevity, $p_{ry} = p_{ry|Y_t}$ and $q_{ry} = q_{ry|Y_t}$ is henceforth used.

As with any MDP, at each time step the state determines the set of allowable controls. Here the decision is a function of the remaining weapons and the current set of targets in the threat environment. For any epoch, $\mathcal{A}_{S_t}$ represents the set of allowable decisions given the system is in state S at time t. Define the decision variables a_{tryj} as the number of weapons of type r to assign to target j, of type y, at time t. A matrix of decisions and the constraint set can be defined as

$$a_t(S_t) = \begin{bmatrix} a_{t111} & a_{t211} & \dots & a_{tM11} \\ a_{t112} & a_{t212} & \dots & a_{tM12} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t11Y_{t1}} & a_{t21Y_{t1}} & \dots & a_{tM1Y_{t1}} \\ a_{t121} & a_{t221} & \dots & a_{tM21} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t12Y_{t2}} & a_{t22Y_{t2}} & \dots & a_{tM2Y_{t2}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t1NY_{tN}} & a_{t2NY_{tN}} & \dots & a_{tMNY_{tN}} \end{bmatrix} \tag{5.2}$$

and

$$\mathcal{A}_{S_t} = \left\{ a(S_t) | \sum_{t=1}^{T} \sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} a_{tryj} \leq R_{1r} \text{ for } r = 1, 2, \dots, M; a_{tryj} \in \mathbb{N} \right\} \tag{5.3}$$

Here the 0 index represents the allowable control of "*do nothing*". At each time step, given a state S_t, action a_t, and outcome W_{t+1}, the system transitions according to

$$S_{t+1} = S^M(S_t, a_t, W_{t+1}) \tag{5.4}$$

where $S^M(\cdot)$ is a function describing the system's dynamics. For our DWTA problem, states transition in two distinct fashions. First, let

$$(a_{tr})_{r=1}^{M} = (\sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} a_{tryj}) \tag{5.5}$$

be a vector denoting the number of weapons of type r fired at time t. Then our weapon state transitions deterministically following

$$R_{t+1} = (R_{tr} - a_{tr})_{r=1}^{M} \tag{5.6}$$

The target vector transitions probabilistically based upon the allocation policy at each decision epoch. Let $\hat{Y}_{t+1,yj}$ be a random variable representing the outcome of the j^{th} target of type y given a decision such that

$$\hat{Y}_{t+1,yj} = \begin{cases} 0 & \text{if target } j \text{ survives the attack,} \\ 1 & \text{if target } j \text{ is destroyed during the attack.} \end{cases} \tag{5.7}$$

for each target type y. Further, define

$$\hat{Y}_{t+1} = \begin{bmatrix} \hat{Y}_{t+1,11} \\ \hat{Y}_{t+1,12} \\ \vdots \\ \hat{Y}_{t+1,1Y_{t1}} \\ \hat{Y}_{t+1,21} \\ \hat{Y}_{t+1,22} \\ \vdots \\ \hat{Y}_{t+1,2Y_{t2}} \\ \vdots \\ \hat{Y}_{t+1,N1} \\ \hat{Y}_{t+1,N2} \\ \vdots \\ \hat{Y}_{t+1,NY_{tN}} \end{bmatrix} \tag{5.8}$$

then the target state element transitions following

$$Y_{t+1} = \left[Y_{ty} - \sum_{j=1}^{Y_{ty}} \hat{Y}_{t+1,yj} \right]_{y=1}^{N} \tag{5.9}$$

and

$$P\{Y_{t+1,yj} = 0 | S_t, a_t\} = \begin{cases} 1 - \prod_{r=1}^{M} (q_{rj})^{a_{trj}} & \text{if } Y_{t,yj} = 1 \\ 1 & \text{if } Y_{t,yj} = 0 \end{cases} \tag{5.10}$$

$$P\{Y_{t+1,yj} = 1 | S_t, a_t\} = \begin{cases} \prod_{r=1}^{M} (q_{rj})^{a_{trj}} & \text{if } Y_{t,yj} = 1 \\ 0 & \text{if } Y_{t,yj} = 0 \end{cases} \tag{5.11}$$

Here, q_{rj} represents the single shot survival probability if weapon type r is shot at target j. This must be done for all active targets with weapons allocated to them at

time t. If n_t denotes the number of active targets with weapons allocated to them at stage t, then if $\hat{\mathcal{Y}}_{t+1}$ is the set of possible outcomes known by time $t+1$, $|\hat{\mathcal{Y}}_{t+1}| = 2^{n_t}$.

As previously discussed, each target has an associated value, V_j. Then the value obtained at any time step follows

$$C_{t+1}(S_t, a_t, \hat{Y}_{t+1}) = \sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} V_y \hat{Y}_{t+1,yj} \tag{5.12}$$

The value of any target destroyed during the time interval $(t, t+1)$ is accumulated within the cumulative objective function value. The objective is determine a policy $\pi \in \Pi$ mapping each state to an action which maximizes

$$\max_{\pi} \mathbb{E}^{\pi} \left\{ \sum_{t \in \mathcal{T}} \gamma C_t^{\pi}(S_t, A_t^{\pi}(S_t)) \right\}. \tag{5.13}$$

where Π is the set of all possible policies and γ is the discount factor.

5.4 Solution Methodology

The SWTA problem has been shown to be NP-complete by Lloyd and Witsenhausen [60], therefore, any extension is also NP-complete. As conditional kill probabilities are incorporated , the sequence in which weapons are employed becomes an important factor. The proposed solution to this problem uses approximate dynamic programming (ADP). Section 5.4.1 introduces dynamic programming and lays the foundation for solution using ADP. Section 5.4.2 provides a description of the approximations used.

5.4.1 Dynamic Programming.

5.4.1.1 Value Iteration.

Dynamic programming is a well demonstrated method for solving MDPs such as the one formulated in 5.3.2. At each time step, t, the value of being in each state is computed using Bellman's equations:

$$J_t(S_t) = \max_{a_t \in A_t} \left(C_t(S_t, a_t) + \gamma \mathbb{E} \left\{ J_{t+1}(S_{t+1}) | S_t \right\} \right) \tag{5.14}$$

$$= \max_{a_t \in A_t} \left(C_t(S_t, a_t) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|S_t, a_t) J_{t+1}(s') \right). \tag{5.15}$$

where the state transitions according to equation 6.7. In order to solve the problem the Gauss-Seidel variant of value iteration is used. This algorithm is

Algorithm 5 Gauss-Seidel Value Iteration Algorithm

1: Initialize: Set $J^0(s) = 0 \ \forall s \in S.$, Fix $\epsilon > 0$, Set $n = 1$.
2: For each $s \in S$ compute:

$$J^n(s) = \max_{a \in \mathcal{A}} \left\{ C(S, a) + \gamma \left(\sum_{s' < s} \mathbb{P}(s'|s, a) J^n(s') + \sum_{s' \geq s} \mathbb{P}(s'|s, a) J^{n-1}(s') \right) \right\} \tag{5.16}$$

3: If $||J^n - J^{n-1}|| < \epsilon(1-\gamma)/2\gamma$, let π^ϵ be the resulting policy that solves 5.22, and let $J^\epsilon = J^n$ and stop; otherwise, set $n = n + 1$ and go to 2.

5.4.1.2 Approximate Dynamic Programming.

Approximate dynamic programming is a technique often used for solving high dimensional resource allocation problems. Many applications exist within the transportation industry [80][81][84] [82]. Further, ADP has been applied to sensor management [21], multiplatform path planning [77], and stochastic scheduling [14]. Another

area which has a significant amount of literature is vehicle routing with stochastic demands [73][86][87][3]. Other resource allocations applications include activity networks for project planning [32][93], model predictive control [22], and high-dimensional generalized resource allocation [81], among others.

The difficulty with practically sized resource allocation problems is that they typically grow exponentially in the state, action, or outcome spaces; the presented DWTA problem is no different. Specifically, for this problem, the decision space grows exponentially as a function of the state space. To illustrate this, assume an arbitrary state S_t where there are m_t weapons remaining and n_t targets. There are then $(n_t + 1)^{m_t}$ possible actions over which the algorithm must iterate. Much of the focus of approximate dynamic programming is to step forward making use of an estimate for the future value of our states. Instead of looping over all states and actions in exact value iteration, this research proposes a reduction of the decision space using the principles of ordinal optimization.

5.4.2 Value Iteration Using a Reduced Decision Space.

Because of the large number of allocations for any state action pair, the use of order statistics is proposed to reduce the size of the action space investigated during value iteration. First, consider the method used by Ho and Sreenivas to optimize discrete event dynamic systems [46] known as ordinal optimization. The purpose of this is to, for each state, select a subset of decisions to investigate such that the best decision from the selected subset is better than a pre-defined population percentile. Ordinal optimization has been used in a wide range of simulation optimization problems to effectively generate high quality solutions. Guan, Ho, and Lai [41] use ordinal optimization to select a set of approximated bidding strategies for electrical power suppliers. After the subset of options is selected, an exact solution is solved, and the

best bidding strategy is selected. Xie, Zhong, and Wu [100] apply a similar approach to the strengthening of transmission networks through the selection of several alternatives, for which more detailed simulations are explored prior to final selection. This research uses ordinal optimization to select a subset of decisions for each state, such that the probability of selecting a *good enough* decision can be fixed. Let $\tilde{\mathcal{A}}_{S_t} \subset \mathcal{A}_{S_t}$ and $|\tilde{\mathcal{A}}_{S_t}| = K$. Each action $a_t \in \tilde{\mathcal{A}}_{S_t}$ has a subsequent expected future reward determined using

$$C_{t+1}(S_t, a_t) = \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|S_t, a_t) J_{t+1}(s') \tag{5.17}$$

Therefore, considering only the subset $\tilde{\mathcal{A}}_{S_t}$, order the samples such that $C_{t+1}^{[1]} < C_{t+1}^{[2]} < \ldots < C_{t+1}^{[K]}$ the largest order from the sample will be the value which maximizes

$$J_t(S_t) = \max_{\tilde{a}_t \in \tilde{\mathcal{A}}_{S_t}} \left(C_t(S_t, \tilde{a}_t) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|S_t, \tilde{a}_t) J_{t+1}(s') \right) \tag{5.18}$$

Define φ such that $0 < \varphi < 1$ as a population percentile and define ρ such that $0 < \rho < 1$ as a desired level of confidence. Then, distribution free confidence intervals are derived for these percentiles so long as our cumulative density function $\Phi(a)$ is strictly increasing because $\Phi(a) = \varphi$ has a unique solution, defined as ξ_φ. Next, select a sample size K which guarantees

$$P\left\{ C_{t+1}^{[K]} > \xi_\varphi \right\} > \rho, \tag{5.19}$$

However,

$$P\left\{ C_{t+1}^{[K]} > \xi_\varphi \right\} = 1 - P\left\{ C_{t+1}^{[K]} < \xi_\varphi \right\} = 1 - \varphi^K, \tag{5.20}$$

which results in

$$1 - \varphi^K > \rho \Rightarrow K \geq \left\lceil \frac{\log(1-\rho)}{\log(\varphi)} \right\rceil. \tag{5.21}$$

This states that, if K samples are selected from any population, with ρ confidence, φ percent of the population would be below the largest order statistic $C_{t+1}^{[K]}$. This principle is used to reduce the number of actions investigated in value iteratioin, and with intelligent alteration of the distribution from which our samples are selected, results in high-quality solutions in much faster computation time. The algorithm is described in Algorithm 6.

Algorithm 6 Gauss-Seidel value iteration with a reduced decision space

1: Initialize: Set $v^0(s) = 0 \; \forall s \in S.$, Fix $\epsilon > 0$, Set $n = 1$.
2: For each $S \in \mathcal{S}$
3: **if** $|A_S| > K$ **then**,
4: Generate a subset of decisions $\tilde{A}_S \subset A_S$ where $|\tilde{A}_S| = K$ according to $\Phi(a)$.
5: **else**
6: $\tilde{A}_S = A_S$
7: **end if**
8: Compute:

$$v^n(s) = \max_{a \in \tilde{\mathcal{A}}_S} \left\{ C(S,a) + \gamma \left(\sum_{s'<s} \mathbb{P}(s'|s,a) v^n(s') + \sum_{s' \geq s} \mathbb{P}(s'|s,a) v^{n-1}(s') \right) \right\} \tag{5.22}$$

9: If $||v^n - v^{n-1}|| < \epsilon(1-\gamma)/2\gamma$, let π^ϵ be the resulting policy that solves 5.22, and let $v^\epsilon = v^n$ and stop; otherwise, set $n = n + 1$ and go to 2.

5.5 Numeric Results

We begin by defining and solving a simple example to illustrate the computational complexity of the problem. Numeric results for this simple example are presented, to include sensitivity analysis and the impact of parametric changes. Finally, the problem is extended to that of a more practical size similar results are discussed.

5.5.1 Simple Example Description.

For this example notional future weapons concepts are investigted, each having different capabilities. The initial state conditions are $M = 5$, $N = 3$, $m = 7$, and $n = 4$. Let

$$R_0 = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 2 \end{pmatrix} \tag{5.23}$$

and

$$Y_0 = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \#\text{SAM target type} \\ \#\text{Radar target type} \\ \#C^2 \text{ target type} \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \tag{5.24}$$

Then, the initial state vector is $S_0 = (R_0, Y_0)$ meaning there are two weapons of type 1, 3 and 5, one weapon of type 4, and 0 weapons of type 2. Target type y is valued according to $V = (V_1, V_2, V_3)^T = (100, 200, 300)^T$. Target state transitions are based on Table 8.

Table 8. Conditional probabilities of the state transitions

	Single Shot p_kill (all target types remain)			No SAMs Remaining		No Radars Remaining		No SAM or Radar
Weapon Type	SAM	Radar	C^2	Radar	C^2	SAM	C^2	C^2
1	0.8	0.6	0.5	0.65	0.6	0.95	0.55	0.6
2	0.6	0.8	0.5	0.9	0.55	0.65	0.55	0.6
3	0.6	0.5	0.8	0.6	0.95	0.65	0.85	0.95
4	0.45	0.6	0.375	0.675	0.4125	0.4875	0.4125	0.45
5	0.45	0.375	0.6	0.45	0.7125	0.4875	0.6375	0.7125

Notice that the single shot probabilities of kill are conditional probabilities that change based upon the targets currently in the threat environment. This is used

to model a scenario in which elimination of a certain target type may degrade the adversarial capability, thus increase the probability of destroying a terminal target. Here the terminal target is a command and control (C^2) target, and the terminal states are when all weapons have been used, or the C^2 target has been destroyed. Though it is assumed the platform from which weapons are fired is out of threat range, risk is implicitly added using a discounting factor, γ.

5.5.2 Simple Example Solution.

A walkthrough for the solution of our simple example is presented along with a brief discussion of the implications of the problem formulation. Using value iteration, all possible states and decisions must be considered. For this simple example with seven weapons and four targets, there are 864 states, of which 556 may be transitioned to feasibly. The computational issue faced is the number of possible decisions when there are many weapons and targets remaining. For the initial state $(n+1)^m = (4+1)^7 = 78,125$ possible decisions must be investigated. The optimal action at $t = 0$ is

$$a^*(S_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this representation, columns reference weapon type, r and rows reference a specific target j, $j = 1, 2, \ldots, n$. The optimal action is to fire one of the first weapon type (r = 1) at each of the active SAMs ($y = 1$), and the only weapon of type $r = 4$ at the Radar. Note that because of the homogeneity of the SAM targets it would be

optimal to alternate which weapons to fire at them while remaining optimal. Using the data in Table 8 and the target values, the expected single-stage contribution is

$$C_0(S_0, A_0^*) = 0.8 * 100 + 0.45 * 100 + (0.6) * 200 = 280 \tag{5.25}$$

Recall the weapon state transitions deterministically, so

$$R_1 = (R_{0r} - a_{0r})_{r \in \mathcal{R}} = \begin{pmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 2 \end{pmatrix} \tag{5.26}$$

The target state, however, would transition to one of six possible target states:

$$Y_1 \in \left\{ \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\} \tag{5.27}$$

Note that $Y_1 = (1, 1, 1)^T$ and $Y_1 = (1, 0, 1)^T$ could each be reached by two different paths, so caution must be taken when computing their probabilities. For the next step, assume $Y_1 = (1, 0, 1)^T$, meaning one of the SAMs and the RADAR were each destroyed and kill probabilities transition. The optimal policy for S_1 is

$$a^*(S_1) = \begin{pmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.28}$$

meaning both weapons of type one are allocated to the remaining SAM target. As a means of validation, it is expected that $V^*(S_1)$ and $A^*(S_1)$ would be the same regardless of which SAM is remaining in the threat environment. This is confirmed in the results. The state will now transition to one of two states:

$$S_2 \in \left\{ \left[\begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]; \left[\begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right] \right\} \tag{5.29}$$

where the obvious optimal decision will be to fire the remaining weapons at the C^2 node, at which point the system is guaranteed to transition to a terminal state. An item of interest comes in looking at the single shot kill probabilities for the different scenarios. When no radars are present, notice $p_{1,1} = 0.95, p_{5,1} = 0.4875, p_{1,4} = 0.55, p_{5,4} = 0.6375$, however, when there's only the C^2 target remaining the probabilities shift to $p_{1,4} = 0.6$ and $p_{5,4} = 0.7125$. It is much more advantageous to shoot the remaining weapon of type one at the SAM because of both the value gained, and the likelihood that the system will transition to a state where only the C^2 remains.

5.5.3 Numeric results for the simple example.

Given an exact solution for our model, numerical comparisons are presented for the approximation techniques. Using the property from Section 5.4.2 that $\Phi(a)$ must be strictly increasing several discrete distributions are selected that determine how a subset of actions are selected for each state. Let $k(a_t)$ be the number of weapons to fire, these distributions are used to determine the number allocations are generated for each $k(a_t)$, which will be some proportion of K.

5.5.3.1 Uniform Discrete Distribution.

As an initial choice, a uniform discrete distribution is used, with CDF

$$\Phi(k(a_t); m_{min}, m_s) = \frac{\lfloor k(a_t) \rfloor - m_{min} + 1}{m_{s_t} - m_{min} + 1} \tag{5.30}$$

where m_{min} is the minimum number of weapons to fire (for our example $m_{min} = 1$), m_s is the number of weapons given the state $s \in S$, and $k(a_t) \in [m_{min}, m_s]$. This method provides a broad exploration of the allocation space.

5.5.3.2 Binomial Distribution.

Next problem knowledge is used to generate discrete distributions which increases the likelihood of selecting good actions. For some initial state, unless the discount factor is low enough, it will be suboptimal to fire all remaining weapons at once. Similarly, it is likely that firing one weapon during a stage where numerous weapons remain in inventory may not be optimal. Additionally, because of the combinatoric nature of the problem, there are a greater number of possible ways of allocating weapons if $k(t)$ does not lie on the bounds of $[m_{min}, m_s]$. Therefore a binomial distribution is used to generate allocations centered around the median number of weapons in each state.

$$\Phi(k(a_t), m_s, p) = \sum_{i=0}^{\lfloor k(a_t) \rfloor} \binom{m_s}{i} p^i (1-p)^{n-i} \tag{5.31}$$

By fixing the sample size at the number of weapons in the state, the success probability parameter is altered to vary the shape of the distribution. The frequencies from our distribution are then used multiplied by K following

$$\hat{k}(a_t) = \lceil \phi(k(a_t)))K \rceil \tag{5.32}$$

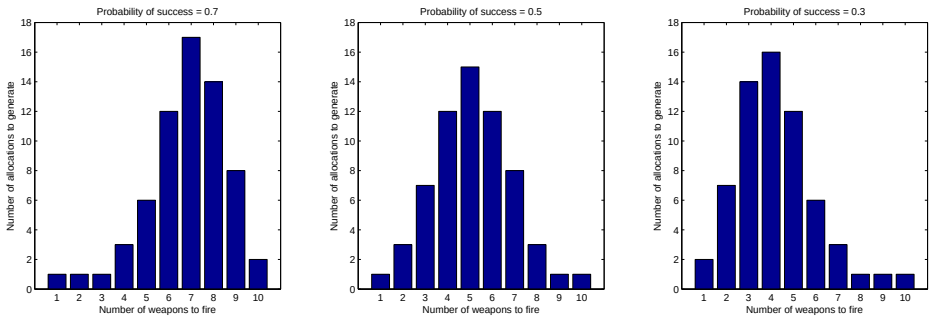


Figure 8. Binomial Selection Distributions for $\varphi = \rho = 0.95 \Rightarrow K \geq 59$

where ϕ represents the binomial probability mass function and $\hat{k}(a_t)$ is the actual number of allocations to generate for $k(a_t)$. The ceiling function guarantees that the actual number of samples will be greater than or equal to K. Three examples for $K = 59$ are presented in Figure 8.

5.5.3.3 Comparative Results.

A comparison of results for the various approximation schemes is now presented for the small example. The probability paramater ϕ is varied for the binomial distribution to see its impact on solution quality. Similarly, φ and ρ are varied. The results are shown in Tables 9-12.

Table 9. Results for $\varphi = \rho = 0.95 \Rightarrow K \geq 59$

	Exact	$UNIF(1, m_s)$	$B(m_s, 0.7)$	$B(m_s, 0.6)$	$B(m_s, 0.5)$	$B(m_s, 0.4)$	$B(m_s, 0.3)$
J^*	636.582	613.825	624.606	629.017	630.153	628.434	627.406
ΔJ^*	-	22.757 ±9.844	11.976 ±8.834	7.565 ±5.238	6.429 ±3.598	8.147 ±4.205	9.176 ±4.375
$\%\Delta J^*$	-	3.6% ±1.5%	1.3% ±1.4%	1.3% ±0.8%	1.3% ±0.6%	1.3% ±0.7%	1.4% ±0.7%
Average Worst $\Delta J(S)$ of all states	-	51.4309 ±9.2023	34.269 ±6.3033	25.7421 ±6.1891	27.0920 ±5.775	24.6047 ±5.4376	25.6585 ±4.459
Average Worst ΔJ % of all states	-	9.7% ±2%	7.1% ±1.9%	5.6% ±1.7%	6.3% ±1.8%	5.8% ±1.6%	6.5% ±1.6%
Average ΔJ $\forall S$	-	1.7367 ±0.1625	1.1106 ±0.1441	0.9213 ±0.0887	0.9879 ±0.1130	1.1178 ±0.1166	1.3604 ±0.0998
Average ΔJ $\forall S$ %	-	0.3% ± < 0.0001	0.2% ± < 0.0001	0.1% ± < 0.0001	0.2% ± < 0.0001	0.2% ± < 0.0001	0.2% ± < 0.0001
Comp Time (sec)	5.952 ±0.0227	0.3109 ±0.0015	0.3122 ±0.0019	0.3097 ±0.0022	0.3100 ±0.0011	0.3107 ±0.0015	0.3035 ±0.0017

Table 10. Results for $\varphi = .95\rho = 0.99 \Rightarrow K \geq 90$

	Exact	$UNIF(1, m_s)$	$B(m_s, 0.7)$	$B(m_s, 0.6)$	$B(m_s, 0.5)$	$B(m_s, 0.4)$	$B(m_s, 0.3)$
J^*	636.582	618.235	627.842	631.346	630.912	631.583	630.474
ΔJ^*	18.347	8.740	5.236	5.670	4.999	6.108	
		±8.944	±5.351	±4.324	±3.774	±3.377	±3.438
$\%\Delta J^*$	-	2.9 %	0.8 %	0.8 %	0.8 %	0.8 %	1.0 %
		±1.4%	±0.8%	±0.7%	±0.6%	±0.5%	±0.5%
Average Worst ΔJ	-	42.1246	26.6107	20.8569	20.2236	23.1063	22.0201
		±7.1965	±4.584	±4.6305	±4.9219	±5.6322	±5.9783
Average Worst ΔJ %	-	8.2%	5.2%	4.6%	4.7%	5.7%	5.6%
		±1.5%	±1.3%	±1.5%	±1.6%	±1.8%	±1.8%
Average $\Delta J \forall S$	-	1.1065	0.6556	0.5364	0.5382	0.5958	0.7739
	±0.1230	±0.0849	±0.0826	±0.0630	±0.0684	±0.0702	
Average $\Delta J \forall S$ %	-	0.17%	0.1%	0.08%	0.08%	0.09%	0.12%
		$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$
Comp Time (sec)	5.952	0.4084	0.4132	0.4121	0.4128	0.4105	0.4123
	±0.0227	±0.0021	±0.0021	±0.0026	±0.0024	±0.0014	±0.0015

Table 11. Results for $\varphi = 0.99\rho = 0.95 \Rightarrow K \geq 299$

	Exact	$UNIF(1, m_s)$	$B(m_s, 0.7)$	$B(m_s, 0.6)$	$B(m_s, 0.5)$	$B(m_s, 0.4)$	$B(m_s, 0.3)$
J^*	636.582	628.943	633.115	634.9849	635.0977	634.2314	634.3026
ΔJ^*	-	7.639	3.467	1.597	1.484	2.351	2.279
		±4.611	±3.482	±1.812	±1.572	±2.55	±2.35
$\%\Delta J^*$	-	1.2%	0.37%	0.37%	0.37%	0.37%	0.36%
		±0.07%	±0.05%	±0.03%	±0.02%	±0.04%	±0.037%
Average Worst ΔJ	-	19.0955	11.9	8.9806	8.7924	7.7779	10.4195
		±3.5135	±2.8815	±3.1484	±2.6031	±1.513	±3.5743
Average Worst ΔJ %	-	3.49%	2.24%	1.7%	1.7%	1.45%	2.1%
		±0.687%	±0.61%	±0.67%	±0.62%	±0.413%	±0.96%
Average $\Delta J \forall S$	-	0.3013	0.1574	0.1087	0.1115	0.1158	0.1510
		±0.0637	±0.0337	±0.0268	±0.0176	±0.0197	±0.0205
Average $\Delta J \forall S$ %	-	0.047%	0.025%	0.017%	0.018%	0.024%	-
		$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$	$\pm < 0.0001$
Comp Time (sec)	5.952	0.8788	0.8800	0.8792	0.8814	0.8778	0.8782
	±0.0227	±0.0036	±0.0033	±0.0047	±0.0030	±0.0031	±0.0033

Table 12. Results for $\varphi = \rho = 0.99 \Rightarrow K \geq 459$

	Exact	$UNIF(1, m_s)$	$B(m_s, 0.7)$	$B(m_s, 0.6)$	$B(m_s, 0.5)$	$B(m_s, 0.4)$	$B(m_s, 0.3)$
J^*	636.582	631.74	634.8	635.37	635.58	634.59	635.26
ΔJ^*	-	4.842 ±4.121	1.782 ±2.34.	1.214 ±1.74	1 ±1.474	1.99 ±1.813	1.324 ±1.41
$\%\Delta J^*$	-	0.76% ±0.65%	0.3% ±0.37%	0.3% ±0.27%	0.3% ±0.23%	0.3% ±0.285%	0.3% ±0.22%
Average Worst ΔJ	-	16.2025 ±3.5294	9.7996 ±3.7777	6.9378 ±1.1341	6.9352 ±0.8392	7.9083 ±2.9455	8.9068 ±3.0443
Average Worst ΔJ %	2.98%	1.83% ±3.53	1.31% ±3.78	1.34% ±1.13	1.55% ±0.84	1.76% ±2.95	- ±3.04
Average $\Delta J \forall S$	-	0.2 ±0.0349	0.1009 ±0.0283	0.0703 ±0.0171	0.0675 ±0.0130	0.0863 ±0.0150	0.1138 ±0.0168
Average $\Delta J \forall S$ %	-	0.005% ± < 0.0001	0.004% ± < 0.0001	0.003% ± < 0.0001	0.002% ± < 0.0001	0.002% ± < 0.0001	0.003% ± < 0.0001
Comp Time (sec)	5.952 ±0.0227	1.0924 ±0.0049	1.0933 ±0.0049	1.0896 ±0.0057	1.0920 ±0.0035	1.0892 ±0.0042	1.0911 ±0.0049

The proposed methods demonstrate a distinct reduction in computation time, compared to the exact method, for each level of K, with a comparatively small optimality gap. All experiments were performed using MATLAB 2013a on an Intel XEON X5667 with 24GB RAM. Note that the binomial distribution performs better than the uniform distribution in all cases. This is not surprising because the shape of the binomial distribution should reinforce the selection of decisions which are more likely to increase the long-term objective value. Additionally, within the family of binomial distributions, $B(10, 0.6)$ or $B(10, 0.5)$ consistently provide better solutions in these runs. Intuitively, solution quality increases as K increases, though not necessarily in a linear manner. Figure 9 shows that we get greater improvement in average ΔJ (for all states) going from $K = 59$ to $K = 90$ in relation to computation time. Further, computation time appears fairly linear with respect to K

5.5.4 Sensitivity Analysis.

Next, sensitivity analysis is performed on various parameters within our problem. First, the impact with which the discount factor γ has on the optimal policy and value function is investigated. The optimal myopic policy for the small example is

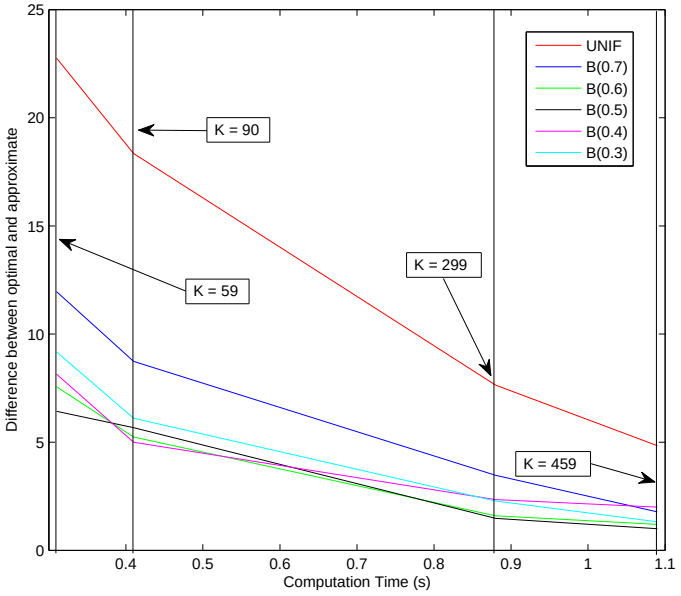


Figure 9. $J^* - \tilde{J}^*$ **by computation time**

$$a^*_{myopic}(S_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

with $J^*_{myopic} = 616.75$. The values of γ are varied and show the dynamics of the system become significant around $\gamma = 0.9$ where myopic is no longer optimal. Recall that $1 - \gamma$ represents the probability that the platforms are "shot down", meaning if the likelihood of being destroyed is $> 10\%$, a static policy will be optimal for the problem defined.

Next a methodology for generating problem instances is presented for further analysis. Two additional weapon types are added to the problem, along with an additional target type, representing a second type of SAM. Because of the sensitivity of actual data, a means for computing kill probabilities with practical significance

Table 13. List of events for defining probability constraints

Event	Active Target Types
E	SAM 1, SAM 2, Radar, C^2
F	Radar, C^2
G	SAM 1 and/or SAM 2, C^2
H	C^2

was needed. First, bounds are set for the probability of kill for each weapon and target type. Define p_{ry}^l and p_{ry}^u as the lower and upper bounds, respectively. Next, constraints are imposed on future kill probabilities as follows. Targets are labeled a, b, c, and d for SAM 1, SAM 2, Radar, and C^2 respectively. Additionally, define the following events; let E denote the event that all target types remain, F denote the event that all SAM targets have been destroyed, G denote the event where all Radar targets have been destroyed, and H denote the event where all SAM and Radar targets have been destroyed. These events are shown in Table 13.

The conditional probability constraints are then

$$p_{rc|F} \geq p_{rc|E}, \tag{5.33}$$

$$p_{rd|F} \geq p_{rd|E}, \tag{5.34}$$

$$p_{ra|G} \geq p_{ra|E}, \tag{5.35}$$

$$p_{ra|G} \geq p_{ra|E}, \tag{5.36}$$

$$p_{rd|G} \geq p_{rd|F}, \tag{5.37}$$

$$p_{rd|H} \geq p_{rd|F}, \tag{5.38}$$

$$p_{rd|H} \geq p_{rd|G}. \tag{5.39}$$

A nearly orthogonal latin hypercube (NOLH) design for up to seven factors was used to generate $p_{ry|E}$ for all r and y. Here the factors are the target types, with

the weapons capabilities denoting the design space to be investigated. This resulted in 17 potential weapons choices from which M are selected randomly according to a uniform distribution. Next for each weapon type r, and active target type y (based upon the specific event), and event F, G, and H, compute the following

$$p_{ry|F} = rand * (p^u_{ry} - p_{ry|E}) + p_{ry|E}, \tag{5.40}$$

$$p_{ry|G} = rand * (p^u_{ry} - p_{ry|F}) + p_{ry|E}, \tag{5.41}$$

$$p_{ry|H} = rand * (p^u_{ry} - p_{ry|G}) + p_{ry|G}. \tag{5.42}$$

This provides exploration of well spaced alternatives for a, b, c, and d and probabilities that satisfy (5.33) - (5.39). It is assumed erroneous to say weapons capabilities would degrade as targets are eliminated from the threat environment. Therefore, as threats are diminished, weapons' capabilities increase in turn. Note that kill probabilities are modeled to implicitly factor in both effectiveness of weapons and the risk that a weapon is shot down during employment. An example matrix which is used for additional analysis is provided in Table 21.

Table 14. Updated conditional transition probabilities

	Single Shot p_{ry} (all target types remain)				No SAMs Remaining		No Radars Remaining			No SAM or Radar
Weapon Type	SAM 1	SAM 2	Radar	C^2	Radar	C^2	SAM 1	SAM 2	C^2	C^2
1	0.47	0.51	0.6	0.59	0.67	0.69	0.75	0.65	0.76	0.76
2	0.53	0.68	0.58	0.54	0.65	0.83	0.87	0.7	0.84	0.92
3	0.48	0.56	0.47	0.51	0.58	0.89	0.86	0.94	0.86	0.91
4	0.55	0.58	0.48	0.56	0.7	0.93	0.68	0.64	0.88	0.94
5	0.47	0.65	0.45	0.62	0.83	0.78	0.54	0.71	0.82	0.83
6	0.56	0.48	0.51	0.47	0.74	0.77	0.55	0.78	0.84	0.9
7	0.64	0.51	0.56	0.48	0.65	0.95	0.7	0.68	0.94	0.95

The analysis was re-run using the probabilities in Table 21, the results are presented in Tables 18-17. This example investigates one target of each type. Based on the results of the initial experiments, this analysis is only performed for for $B(m_s, 0.7), B(m_s, 0.5)$, and $B(m_s, 0.3)$. This provides a spread of binomial distri-

butions for comparison, but excludes approximation using the uniform distribution due to its relative poor performance. Additionally, computation time for the next set of experiments was almost identical to the initial experiments, and is omitted from the results.

Table 15. Results for $\varphi = \rho = 0.95 \Rightarrow K \geq 59$ using updated kill probabilities

	Exact	$B(m_s, 0.7)$	$B(m_s, 0.5)$	$B(m_s, 0.3)$
J^*	627.2211	599.0734	612.7626	620.2505
ΔJ^*	-	28.1477 ±10.2858	14.4585 ±7.5289	6.9705 ±2.999
$\%\Delta J^*$	-	4.488% ±1.64%	2.3% ±1.2%	1.1% ±0.48%
Average Worst $\Delta J(S)$ of all states	-	73.5004 ±7.799	44.7731 ±8.4349	24.1433 ±8.3095
Average Worst ΔJ % of all states	-	11.72% ±1.24%	7.14% ±1.34%	3.85% ±1.32%
Average ΔJ $\forall S$	-	3.5588 ±0.2746	1.4147 ±0.1277	0.596 ±0.0981
Average ΔJ $\forall S$ %	-	0.6799% ±0.00044	0.6765% ±0.0002	0.6752% ±0.00016

Table 16. Results for $\varphi = .95\rho = 0.99 \Rightarrow K \geq 90$

	Exact	$B(m_s, 0.7)$	$B(m_s, 0.5)$	$B(m_s, 0.3)$
J^*	627.2211	603.2401	617.1384	621.5328
ΔJ^*	-	23.981 ±9.8287	10.0827 ±5.423	5.6883 ±3.1795
$\%\Delta J^*$	-	3.82% ±1.57%	1.61% ±0.86%	0.91% ±0.51%
Average Worst $\Delta J(S)$ of all states	-	65.4856 ±10.7032	35.7083 ±6.6988	15.9688 ±6.1307
Average Worst ΔJ % of all states	-	10.44% ±1.71%	5.69% ±1.07%	2.55% ±0.98%
Average ΔJ $\forall S$	-	2.4767 ±0.1545	0.8271 ±0.1239	0.2854 ±0.0456
Average ΔJ $\forall S$ %	-	0.6782% ±0.00025	0.6756% ±0.0002	0.6743% ±0.00007

Table 17. Results for $\varphi = 0.99\rho = 0.95 \Rightarrow K \geq 299$

	Exact	$B(m_s, 0.7)$	$B(m_s, 0.5)$	$B(m_s, 0.3)$
J^*	627.2211	614.8137	622.1174	624.61
ΔJ^*	-	12.4074 ±5.9636	5.1037 ±3.6121	2.6111 ±1.4042
$\%\Delta J^*$	-	1.98% ±0.95%	0.81% ±0.58%	0.42% ±0.22%
Average Worst $\Delta J(S)$ of all states	-	38.7502 ±5.1409	16.2579 ±7.4426	4.6529 ±1.5964
Average Worst ΔJ % of all states	-	6.18% ±0.82%	2.59% ±1.19%	0.74% ±0.25%
Average ΔJ $\forall S$	-	0.8467 ±0.1146	0.1637 ±0.0511	0.0409 ±0.0113
Average ΔJ $\forall S$ %	-	0.676% ±0.0002	0.675% ±0.0001	0.674% ±0.00002

Table 18. Results for $\varphi = \rho = 0.99 \Rightarrow K \geq 459$ using updated kill probabilities

	Exact	$B(m_s, 0.7)$	$B(m_s, 0.5)$	$B(m_s, 0.3)$
J^*	627.2211	617.0361	624.6009	625.2398
ΔJ^*	-	10.185 ±5.0288	2.602 ±2.5558	1..9813 ±1.0368
$\%\Delta J^*$	-	1.62% ±0.80%	0.42% ±0.41%	0.32% ±0.17%
Average Worst $\Delta J(S)$ of all states	-	33.6234 ±5.5104	10.1703 ±6.0695	3.6597 ±0.9198
Average Worst ΔJ % of all states	-	5.36% ±0.88%	1.62% ±0.97%	0.58% ±0.15%
Average ΔJ $\forall S$	-	0.6189 ±0.0809	0.0808 ±0.0276	0.026 ±0.0065
Average ΔJ $\forall S$ %	-	0.6752% ±0.00013	0.6744% ±0.00004	0.6743% ±0.00001

The results are fairly consistent with the initial experiments, with a few exceptions. The greatest improvement seen with the second set of experiments comes with a binomial parameter of $\phi = 0.3$, where as the most improvement was previously obtained using the binomial success parameter of $\phi = 0.4, 0.5$, or 0.6. One explanation for this change is that the method reinforces reservation of weapons for future stages given the updated probability tables. With the arbitrarily generated kill probabilities, the sequential destruction of targets was not as necessary because for each stage, weapons which had great effect on different target types were likely present. This early-stage effectiveness may cause the method to fire more weapons earlier.

5.5.5 Numeric results for larger problems.

Using the information gleaned from Section 5.5.3.3 the effectiveness of the proposed method is performed on larger problem instance. Because of the improvement in solution quality with a comparatively small increase in computation time, K is fixed at 59. Additionally, because the greatest improvement in solution quality was obtained using various binomial distributions, they are investigated further in large scale problems. Since the size of the decision space for these problems is so large (a problem with 10 weapons and 10 targets has $|\mathcal{A}_{S_0}| \approx 26B$), comparison with the exact optimal is computationally prohibitive. Instead, a myopic approach is developed. In the myopic approach, the decision space becomes the set of all possible weapons able to be allocated for a given state. Essentially, this means that for any given initial state, $\mathcal{A} = \mathcal{S}$. For each decision, a static weapon-target assignment problem is solved through simple recursion to determine the optimal allocation for the state-action pair while using the dynamic kill probabilities. Because the decision space is much small in this case, exact value iteration can be used. However, the decisions are now myopic due to their single-stage solution. The number of states over which must be iterated is the primary metric in determining adequate problem size. An example with 20 weapons and 20 targets has over seven million states, at which point storage and computation becomes an issue. Therefore, for the demonstrated analysis, the problem size is limited to ten or 12 weapons and seven or ten targets. This limitation also provides some practicality in a geographic sense, as threats which are farther away will likely not be considered in an optimal policy given the problem assumptions. The cases with seven targets have two each of SAM1, SAM2, and radar, with a single C^2 target. The problems with ten targets have four SAM1 targets, three SAM2 targets, two radars, and one C^2. Weapons were arbitrarily selected such that each weapon type had at least one, and the remainder were spread evenly across weapon types.

For the ten weapon problems, $R_0 = (2121121)^T$ and for the tweleve weapon problems, $R_0 = (2221122)^T$. For the ADP method, the success parameter is set to $\phi = 0.4, 0.5,$ and 0.6 and ten replications of each are run. The results are reported in Table 19.

Table 19. Results of large scale experiments

Weapons	Targets	Distribution	# States	J^*_{myopic}	$J^*_{dynamic}$	%Improvement
10	7	$B(m_s, 0.4)$			991.1161 ± 13.9186	$13.5 \pm 1.59\%$
10	7	$B(m_s, 0.5)$	18,816	873.24	967.1818 ± 19.2715	$10.76 \pm 2.21\%$
10	7	$B(m_s, 0.6)$			969.4806 ± 12.5804	$11.02 \pm 1.44\%$
12	7	$B(m_s, 0.4)$			1062.9 ± 6.2844	$10.33 \pm 0.65\%$
12	7	$B(m_s, 0.5)$	46,570	963.39	1049.3 ± 8.0023	$8.92 \pm 0.83\%$
12	7	$B(m_s, 0.6)$			1046.4 ± 8.8097	$8.61 \pm 0.91\%$
10	10	$B(m_s, 0.4)$			1026.7 ± 14.0966	$9.01 \pm 1.5\%$
10	10	$B(m_s, 0.5)$	29,676	941.878	985.1773 ± 17.9595	$4.6 \pm 1.9\%$
10	10	$B(m_s, 0.6)$			975.6688 ± 24.7253	$3.5 \pm 2.63\%$

Table 20. Computation time (in seconds) of large scale experiments

Weapons	Targets	Myopic	ADP
10	7	653.8 ± 2.27	15.4436 ± 0.0758
12	7	$1,586.3 \pm 7.34$	$38.7864 \pm .1587$
10	10	$1,034.2 \pm 4.38$	25.2761 ± 0.1794

As is expected, there is a significant improvement gained in this analysis with the proposed method. By considering the impact current allocations have on the future, the ADP method shows an approximate improvement of 10% over the myopic solution. The large scale problems also suggest further evidence that, given the kill probabilities from Table 21, it is beneficial to reserve more weapons for future stages. Additionally, the proposed method gains validation when considering the binomial distribution with $\phi = 0.6$. Because of the shape of the binomial distribution, more decisions are selected which reinforce the firing of a greater number of weapons at each stage. Firing many weapons early on does not allow for the dynamic kill probabilities to take full effect, and solution quality degrades.

The other benefit of the proposed method is that computation time is small considering the number of states and actions over which are iterated. As can be seen in

Table 20, the ADP method consistently outperforms the exact myopic solution. This is due to the increase in the size of the decision space as problem size increases.

5.6 Conclusions

This chapter presents a new class of weapon-target assignment in which kill probabilities are dependent on the current target set and change over time. An approximate dynamic programming solution method is introduced which incorporates a reduced decision space using the properties of order statistics. This reduced decision space is used to quickly provide high-quality solutions. Several distributions are described to determine how elements from the decision space are selected. Results for the examples tested show that solutions for small scale problems are within 1% of optimal using a small subset of the full decision space. The large scale problems tested also show vast improvement over myopic decision policies.

Future research will include investigation of different approximation dynamic programming techniques. The structure of this problem is such that the size of the decision space is prohibitively large, so methods which address this curse of dimensionality are desired. Though it was slower computationally, implementing a multi-step look ahead solution within the myopic framework may result in better solution quality because it is an exact method in the sense that it iterates over the full state and decision spaces. Additionally, a reduced decision space could be coupled with state reduction techniques such as aggregation to further reduce computation time while maintaining solution quality. Investigating roll-out algorithms which take into account the future impact of current decisions may be implementable within a simulation framework to quickly determine optimal policies for problems of a larger size.

VI. An Integrated Simulation Framework for Optimal Weapons-Mix Determination

6.1 Abstract

Genetic algorithms (GAs) are often used for solving stochastic optimization problems because of their exploratory and exploitative properties. GAs can be powerful tools which effectively search a problem's solution space, but in many cases they have their limitations. If the solution space of the problem to be investigated is too large, GAs may suffer from sub-optimality or slow convergence. Further, if the problem to be optimized is of a black-box nature, global optimality is difficult to prove. This research investigates an embedded optimization framework in which a GA is used to optimize the mix of concept weapons. A knapsack formulation is used to determine the best mix of weapons, with a weapon-target assignment problem used to determine optimal weapons capabilities. The utility of each weapon type is initially unknown and determined through simulated employment. However, because the capabilities, namely the probability of destroying a target given the current target set, of each weapon type are unique, their allocation is dependent on the current mix of weapons being tested. Further, the sequencing and allocation policies also depend on the capabilities of the current weapons' mix. A portion of the gene structure within the GA is dedicated to the sequence or allocation policy in which the weapons are used. This research proposes two solutions to this problem for a GA. First, a gene structure which includes the sequencing of weapons is proposed, and at each stage, a static weapon-target assignment problem is solved optimally to determine the weapons' allocation. As an alternative, a method is proposed which uses approximate dynamic programming (ADP) to determine near optimal allocation strategies in order to reduce the design space searched by the GA. In each case, the fitness function for each

design point, or weapon set, is determined through simulation. Results demonstrate that the ADP method converges in fewer generations than the baseline GA, while the baseline GA converges with less computation time.

6.2 Introduction

Combat simulations provide a means for military analysts to investigate a wide range of problems using fewer resources than testing actual systems. Many scenarios are able to be simulated in which real-world data would not be feasibly attainable. Air Force Research Laboratory (AFRL) analysts are developing an integrated framework which will help investigate the proposed effects of future weapons systems in a variety of scenarios. Part of this effort is to determine synergistic effects of weapons against an integrated air defense system (IADS) and consequently optimize a mix of weapons classes to load onto an aircraft. The previous optimization strategy uses a genetic algorithm (GA) which generates and updates populations of candidate solutions. These candidate solutions are tested by stepping forward and backward through time, randomly selecting allocation policies, simulating engagement outcomes, and continuing on until a terminal state has been realized. Upon success of a simulated mission, a candidate allocation strategy is stored for further testing. This process is repeated for each weapons mix within the GA until a locally optimal strategy has been determined or some other termination criteria has been met.

This chapter introduces methods for the optimal aircraft weaponeering using an embedded optimization framework in order to maximize the damage against a known set of targets. Embedded optimization problems use the optimal solution of one problem in order to optimize a primary objective function.

Because of their complexity, examples of embedded optimization problems are sparsely found in the literature. Some examples are the location of groundwater

systems [9], incorporating chaotic maps for PSO parameter adaptation [6], and the optimization of hydrogen networks [106].

The primary objective function for this embedded optimization problem represents a constrained knapsack problem where the utility of each item depends on the total set of items within the knapsack. A genetic algorithm is developed to search the candidate solutions which are then tested within a simulation to determine their combined utility. Next, a multi-stage dynamic weapon-target assignment (DWTA) problem is solved using approximate dynamic programming (ADP) which generates near optimal sequential allocation strategies for the current set of weapons.

The remainder of the chapter is structured as follows. Section 6.3 introduces the knapsack problem and gives the formal definition for each element, including the DWTA subproblem. The GA methodology is discussed in Section 6.4 where the solution of the DWTA through ADP is developed. Next, numerical results are presented in Section 6.5, and some conclusions and areas for future research are in Section 6.6.

6.3 Problem Formulation

The problem is formulated as a multi-dimensional knapsack problem which represents a mix of weapons loaded on a set of aircraft. The objective for this problem is to optimize the set of weapons such that, when employed against a known set of targets, damage to the targets is maximized. The utilities are determined by solving a dynamic weapon target assignment problem using the existing weapon set. First the multi-dimensional knapsack problem is formally presented.

6.3.1 Multi-dimensional Knapsack Problem.

Let x_{ij} denote the number of weapons of type $i, i = 1, 2, \ldots, M$ to load onto aircraft $j, j = 1, 2, \ldots, N$. Define u_i as the utility, which is considered an effectiveness measure, of weapon type i, c_j as the capacity of aircraft j, and w_i as the size or weight of weapon i. Additionally, let $\mathcal{I}_j$ be the set of weapon types that are able to go on aircraft type j. The objective is then

$$\max_x \sum_{j=1}^{N} \sum_{i=1}^{M} u_i x_{ij} \tag{6.1}$$

subject to

$$\sum_{i=1}^{M} w_i x_{ij} = c_j \text{ for } j = 1, 2, \ldots N \tag{6.2}$$

$$x_{ij} \in \mathbb{N} \text{ if } i \in \mathcal{I}_j, 0 \text{ otherwise.} \tag{6.3}$$

One novelty of this problem is that the utilities are functions of the weapons currently in the solution. Let $\vec{x} = (x_{11}, x_{12}, \ldots, x_{1N}, x_{21}, x_{22}, \ldots x_{2N}, \ldots x_{MN})$ be the current decision vector, and $\vec{u} = (u_1, u_2, \ldots, u_M)$ be the current vector of weapon utility. Then $\vec{u} = f(\vec{x})$, where $f(\cdot)$ is a function defined by the solution to a separate subproblem. For this research, the utilities are based upon the weapons' effects within a dynamic weapon target assignment problem. This problem can be solved exactly, approximately, or even estimated through simulation.

Various methods have been used to solve knapsack problems, from dynamic programming [53] [79] [65], to numerous heuristics such as ant colony optimization (ACO) [55] [92], tabu search [43] [36], and GAs [27] [90]. Additional references can be found

in [53]. Next the DWTA problem is presented as it forms the basis for defining weapons' utilities.

6.3.2 Dynamic Weapon-Target Assignment Problem.

The weapon-target assignment (WTA) problem is a model of combat operations which maximizes the total expected damage caused to the enemy's targets (or minimize the value of leaker missiles) using a limited number of weapons. Optimally assigning interceptors to targets is a subject that has become increasingly important with the proliferation of ballistic missiles. The WTA problem is known to be NP-complete [60]. In general, two cases of the WTA problem are considered, *static* and *dynamic*. The static case concerns itself with n known targets and m known weapon types within a single stage. Optimal solution algorithms are known for two cases of the static WTA (SWTA) problem. These cases are when all the weapons are identical [30] [52] and when the targets can receive at most one weapon [24] [75]. The dynamic case can involve additional stochastic elements, multiple stages and other unique characteristics. While no efficient exact solutions of the generalized SWTA problem exist, much research has been done to effectively determine near optimal allocation policies [42]. Specifically, various heuristics have been applied to include generalized network flow [5], genetic algorithms [59] [99], neural networks [96] and Lagrange relaxation [72].

6.3.2.1 Dynamic Weapon-Target Assignment Problem.

The problem is modeled as an infinite horizon, discrete time Markov decision process (MDP) using the collection of objects

$$\{\mathcal{T}, \mathcal{S}, \mathcal{A}, p(\cdot|S, a), C(S, a, W)\} \tag{6.4}$$

where $\mathcal{T}$ is the set of decision epochs, $\mathcal{S}$ is the state space, $\mathcal{A}_S$ represents the set of allowable actions given the system is in state S, with $\mathcal{A} = \bigcup_{S \in \mathcal{S}} \mathcal{A}_S$, $p(\cdot|S, a)$ is the probability transition function conditioned on being in state S and making decision $a \in \mathcal{A}_S$, and $C(S, a, W)$ is the reward obtained from being in state S, making decision a, and realizing the outcome W.

Let $\mathcal{T} = \{1, 2, \ldots\}$ be the set of time stages and let $t \in \mathcal{T}$ denote a specific stage. Let $S_t = (R_t, Y_t) \in \mathcal{S}$ denote the state of the system at time t, where R_t is a vector indicating the number of weapons (of M different types) remaining in inventory and Y_t is a vector indicating the number of targets (of N different types) still functioning. $R_t = (R_{t1}, R_{t2}, \ldots R_{tM})$, where R_{tr} is the number of weapons of type r at time t, $r = 1, \ldots, M$. $Y_t = (Y_{t1}, Y_{t2}, \ldots Y_{tN})$, where Y_{ty} is the number of targets of type y at time t, each with associated value, V_y, $y = 1, \ldots, N$. A state $S \in \mathcal{S}$ corresponds to a particular pair of vectors indicating the number of weapons and targets remaining. Define $p_{ry|Y_t}$ as the single-shot probability of kill if weapon type r is allocated to target type y given the current target set Y_t. Define $q_{ry|Y_t} = 1 - p_{ry|Y_t}$ as the corresponding probability of survival. The conditional probabilities of survival are used to model the cooperative nature of an IADS; as certain targets are destroyed, the attacker achieves improved probability of destroying other targets. For brevity, $p_{ry} = p_{ry|Y_t}$ and $q_{ry} = q_{ry|Y_t}$ is henceforth used.

As with any MDP, at each time step the state determines the set of allowable controls. The decision is a function of the remaining weapons and the current set of targets in the threat environment. For any epoch, $\mathcal{A}_{S_t}$ represents the set of allowable decisions given the system is in state S at time t. Define the decision variables a_{tryj} as the number of weapons of type r to assign to target j, of type y, at time t. A matrix of decisions and the constraint set can be defined as

$$a(S_t) = \begin{bmatrix} a_{t111} & a_{t211} & \dots & a_{tM11} \\ a_{t112} & a_{t212} & \dots & a_{tM12} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t11Y_{t1}} & a_{t21Y_{t1}} & \dots & a_{tM1Y_{t1}} \\ a_{t121} & a_{221} & \dots & a_{M21} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t12Y_{t2}} & a_{t22Y_{t2}} & \dots & a_{tM2Y_{t2}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t1NY_{tN}} & a_{t2NY_{tN}} & \dots & a_{tMNY_{tN}} \end{bmatrix} \tag{6.5}$$

and

$$\mathcal{A}_{S_t} = \left\{ a(S_t) | \sum_{t=1}^{T} \sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} a_{tryj} \leq R_{1r} \text{ for } r = 1, 2, \dots, M; a_{tryj} \in \mathbb{N} \right\} \tag{6.6}$$

Here the 0 index represents the allowable control of "*do nothing*". At each time step, given a state S_t, action a_t, and outcome W_{t+1}, the system transitions according to

$$S_{t+1} = S^M(S_t, a_t, W_{t+1}) \tag{6.7}$$

where $S^M(\cdot)$ is a function describing the system's dynamics. For the DWTA problem, states transition in two distinct fashions. First, let

$$(a_{tr})_{r=1}^{M} = (\sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} a_{tryj}) \tag{6.8}$$

be a vector denoting the number of weapons of type r fired at time t. Then the weapon state transitions deterministically following

$$R_{t+1} = (R_{tr} - a_{tr})_{r=1}^{M} \tag{6.9}$$

The target vector transitions probabilistically based upon the allocation policy at each decision epoch.

Let $\hat{Y}_{t+1,yj}$ be a random variable representing the outcome of the j^{th} target of type y given a decision such that

$$\hat{Y}_{t+1,yj} = \begin{cases} 0 & \text{if target } j \text{ survives the attack,} \\ 1 & \text{if target } j \text{ is destroyed during the attack.} \end{cases} \tag{6.10}$$

for each target type y. Further, define

$$\hat{Y}_{t+1} = \begin{bmatrix} \hat{Y}_{t+1,11} \\ \hat{Y}_{t+1,12} \\ \vdots \\ \hat{Y}_{t+1,1Y_{t1}} \\ \hat{Y}_{t+1,21} \\ \hat{Y}_{t+1,22} \\ \vdots \\ \hat{Y}_{t+1,2Y_{t2}} \\ \vdots \\ \hat{Y}_{t+1,N1} \\ \hat{Y}_{t+1,N2} \\ \vdots \\ \hat{Y}_{t+1,NY_{tN}} \end{bmatrix} \tag{6.11}$$

then the target state element transitions following

$$Y_{t+1} = \left[Y_{ty} - \sum_{j=1}^{Y_{ty}} \hat{Y}_{t+1,yj} \right]_{y=1}^{N} \tag{6.12}$$

and

$$P\{Y_{t+1,yj} = 0 | S_t, a_t\} = \begin{cases} 1 - \prod_{r=1}^{M} (q_{rj})^{a_{trj}} & \text{if } Y_{t,yj} = 1 \\ 1 & \text{if } Y_{t,yj} = 0 \end{cases} \tag{6.13}$$

$$P\{Y_{t+1,yj} = 1 | S_t, a_t\} = \begin{cases} \prod_{r=1}^{M} (q_{rj})^{a_{trj}} & \text{if } Y_{t,yj} = 1 \\ 0 & \text{if } Y_{t,yj} = 0 \end{cases} \tag{6.14}$$

Here, q_{rj} represents the single shot survival probability if weapon type r is shot at target j. This must be done for all active targets with weapons allocated to them at time t. If n_t denotes the number of active targets with weapons allocated to them at stage t, then if $\hat{\mathcal{Y}}_{t+1}$ is the set of possible outcomes known by time $t+1$, $|\hat{\mathcal{Y}}_{t+1}| = 2^{n_t}$.

As previously discussed, each target has an associated value, V_j. Then the value obtained at any time step follows

$$C_{t+1}(S_t, a_t, \hat{Y}_{t+1}) = \sum_{y=1}^{N} \sum_{j=1}^{Y_{ty}} V_y \hat{Y}_{t+1,yj} \tag{6.15}$$

We accumulate the value of any target destroyed during the time interval $(t, t+1)$. The objective is determine a policy $\pi \in \Pi$ mapping each state to an action which maximizes

$$\max_{\pi} \mathbb{E}^{\pi} \left\{ \sum_{t \in \mathcal{T}} \gamma C_t^{\pi}(S_t, A_t^{\pi}(S_t)) \right\}. \tag{6.16}$$

where Π is the set of all possible policies and γ is the discount factor.

The DWTA problem provides a more practical implementation by including a temporal component. As such, the DWTA is a much more complex problem from a mathematical standpoint which has received a fair amount of attention in the literature. Similar to the SWTA, numerous methods have been employed to provide solutions for various types of DWTA problems. As the originator of the dynamic instance, [47] provides several results which are generalizable to the DWTA problem. [70] and [71] uses stochastic decomposition for the two-stage problem previously defined. An extension of the generalized two-stage problem called the shoot-look-shoot target assignment problem also has a fair amount of associated literature, but as it us fundamentally different, it is not discussed herein. Specific to the general DWTA problem, [24] uses a static WTA approximation scheme within an iterative linear network flow framework to effectively provide high-quality solutions for the DWTA. Because of the integer restriction for the decision variables, the chromosome representation within a GA presents a useful scheme for solving both the static and dynamic versions of the WTA problem. As such, much work has developed hybrid GAs to assist in solving the DWTA. [99] apply a modified GA to the DWTA and introduces weapon use deadlines within the problem formulation. These deadlines follow the principles of scheduling theory, and are in the form of additional constraints such that a weapon has to be shot at a target by a specified time or it is rendered unusable. The authors call their method a modified GA because it applies a basic GA iteratively, assigning a weapon to a target (possibly suboptimally) immediately before the deadline is reached. [101] develop a heuristic which uses problem information (domain knowledge) and constraint programming to assign priorities to assignments. Evolutionary heuristics, which use a hybridized GA with memetic algorithms, have also been applied to the DWTA by [25]. Additionally, [54] applies a hybrid heuristic which uses a simulated annealing (SA) type heuristic to determine the fitness of a

population within a GA framework. Other heuristic techniques applied to the DWTA include Tabu Search [102], ACO with tabu table updates [103], and a modified Hungarian method with PSO [56] (though this is in an open source text, so it's rigor may be unverified). Lastly, exact dynamic programming [91][89] has also been applied to the DWTA. The last portion of the WTA literature review focuses on the specific shoot-look-shoot scenario, as well as some miscellaneous methods which are not explicitly weapon-target assignment problems.

6.4 Methodology

In this section the solution approach is discussed, to include the specific details of the GA, and the near-optimal allocation generation using ADP. Finally, the integrated framework is introduced, and the various algorithms are presented.

6.4.1 Genetic Algorithms.

Because of its complexity and the stochastic nature of the decision variable utilities, achieving an optimal mix of weapon types under constraints may not be efficiently obtained through traditional optimization methods. Because of the ability to specifically design its heuristic characteristics, GAs provide a flexible means for investigating combinatorial optimization problems, especially those with integer solutions. Genetic algorithms are search procedures intended to mimic the natural evolution of biologic systems in which characteristics which provide improvement to the fitness are selected in lieu of those in which quality is not demonstrated. Genetic algorithms have been shown effective in a wide range of resource allocation problems including project scheduling [45] [44], knapsack problems [27] [90], and target assignment [99] [25]. The general steps of a GA are presented in Algorithm 7.

Algorithm 7 General steps of a GA

Initialize

- Generate population P,
- Set parent selection, mutation, and crossover parameters

while number of generations has not been reached **do**

- Determine Fitness of each population member
- Select the parent population for mating
- Generate offspring using crossover rules and parent population
- Ensure feasibility of offspring and correct any infeasibility
- Determine any mutated member(s)

end while

The GA is developed by structuring the gene, computing the fitness function, determining how to select the parent population, and dictating how offspring are generated through crossover and mutation.

6.4.1.1 Gene Structure.

For a knapsack problem with known utilities or value, the gene consists of a string of N integer elements defining a feasible mix of weapons [27]. Because the weapon utility is uncertain, the gene is structured to accommodate allocation information used to solve the DWTA during simulation. Specific elements of the gene are also designated for each aircraft type to ensure the feasibility with constraint (6.2). For the first method, the gene includes a string of integer characters representing the time step in which the weapon is to be fired. Define T as the maximum number of engagement time periods and s_k as the stage in which the k^{th} weapon will be used, $k = 1, 2, \ldots, M_j$ for $j = 1, 2, \ldots, N$, and $s_k \in \{1, 2, \ldots, T\}$. An example is shown in Figure 10 with two aircraft being used. In this example, aircraft one has a capacity of eight, aircraft two has a capacity of two, and $w_i = 1$ for $i = 1, 2, \ldots, N = 7$. For this example, an additional constraint is induced that the four weapon types which can

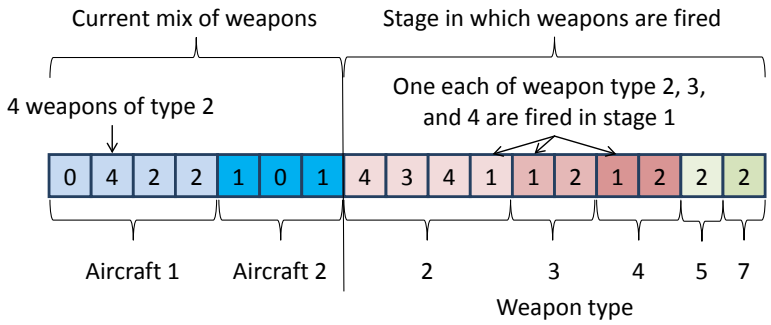


Figure 10. Gene structure for method one

fit on aircraft one cannot fit on aircraft two, and though the three types of weapons are able to be placed on aircraft two, their capabilities are such that they will not be selected for inclusion on aircraft one in an optimal solution. For the example shown, the current gene has zero weapons of type one, four weapons of type two, two weapons of type three and four, and one each of weapon type five and seven. Additionally, this genetic structure provides the stage in which the weapons are to be fired. Set $T = 4$, then one each of weapon type two, three, and four are fired in stage one, followed by one each of weapon type three through seven, and the remaining weapons of type two are fired in stages three and four.

The second method uses the weapon mix portion of the gene structure, but in place of the stage selection, each gene is used to solve a DWTA problem through ADP. The ADP solution methods are from [76] and generate near optimal allocations for any mix of weapons based on the targets represented in the simulation. Figure 11 presents this solution framework.

6.4.1.2 Initial Population.

Similar to the work of Chu [27], the initial population size is set to $P = 50$, and genes are generated randomly. Feasibility of each gene in the initial population is ensured by randomly adding weapons to slots on the aircraft until constraint 6.2

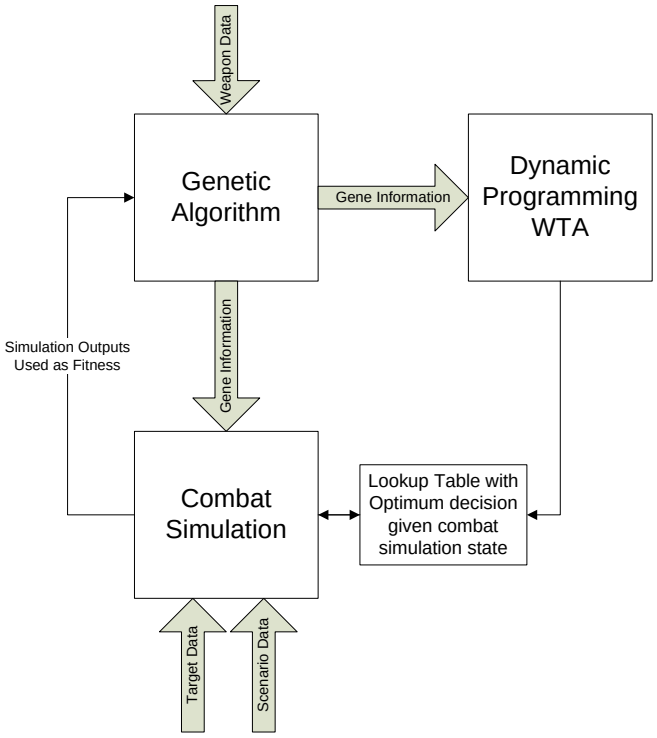


Figure 11. Simulation framework using ADP solution of DWTA

has been satisfied. This operation is performed independently for each aircraft $j = 1, 2, \ldots, N$. Once a feasible gene structure has been generated for each aircraft they are combined to make a full gene. The genes of the initial population are used within the simulation framework to determine their relative fitness before parent selection.

6.4.1.3 Fitness determination.

For each method, a Monte Carlo simulation is used to determine the fitness of the current mix of weapons. For the first method, a static weapon target assignment problem (SWTA) is solved using the weapons fired during a specific stage. Because the number of weapons to be fired at any give stage is generally small, a recursive method is used to optimally generate single stage assignments. The second method, however, uses ADP to solve the DWTA formulated in Section 6.3.2.1 for the current gene. Allocation policies for all possible states are approximated using the methods

discussed in Section 6.4.2. These policies are used as inputs within the Monte Carlo simulation via a lookup table. As the simulation steps through time, allocations are applied based on the DWTA outputs and the outcomes simulated and the state is updated. In each case, 1,000 simulations are run to determine the expected weapons effectiveness against the targets. This average value is then used as the fitness within the GA.

6.4.1.4 Selection of Parent Population.

Parent selection is the determination and assignment of individuals in the population which have comparatively favorable genes which should be passed on to the offspring. Two parents are selected and crossover operators are used to generate offspring. An elitist model is employed within the GA where the top $\zeta\%$ of genes are selected as primary mates. The remainder of the population is divided equally amongst the primary mates to make the next generation. This portion of the population is called secondary mates. The top $P(1-\zeta)/P * \zeta$ are assigned to the top primary mate, the next group of secondary mates are assigned to the second primary mate, and so on. This parental scheme is employed in both GA methods investigated.

6.4.1.5 Crossover and Mutation.

The integer representation of the genetic structure allows for an easy crossover operator implementation. Because GAs are generally insensitive to crossover operator choice [27], the crossover selection is generated randomly based on uniform selection. For method one, a second crossover point is included which is restricted to the sequencing portion of the genetic structure. This allows for better exploration of the design space. This second crossover is also uniformly selected. An example of the crossover operator is shown in Figure 12

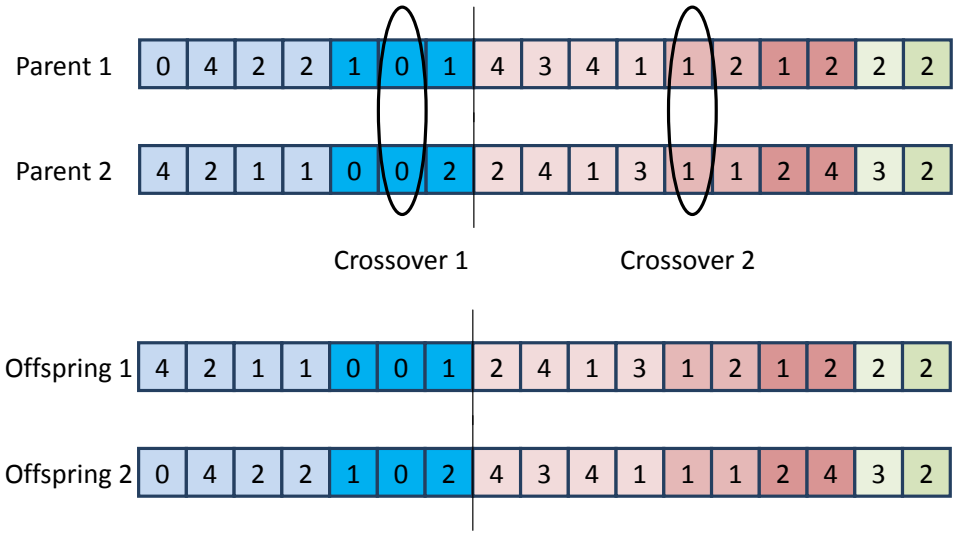


Figure 12. Crossover operator for method one

The crossover operator for method two is restricted to a single crossover point in the weapons mix portion of the gene structure. In this scheme, each set of parents generates two children, so the size of the next generation remains constant. Based on the recommendations of Chu and Beasley [27], these operators were arbitrarily selected, but were kept because computational results were positive. Additionally, a mutation parameter η is implemented to determine if any offspring elements are changed randomly. The mutation probability can help increase or decrease exploration, but is traditionally set to a small value. A random check occurs for each gene, and when applicable, a new value is randomly selected for single element of the weapons mix gene structure.

6.4.1.6 Offspring Feasibility Correction.

In certain cases, the offspring created by crossover and mutation operations are infeasible, because of the equality constraints of (6.2). To guarantee feasibility a repair operator is applied based on the offspring gene and random selection. If the equality constraints are not satisfied randomly selected elements from the gene structure are

either increased or decreased until feasibility is regained. This represents adding or removing weapons from aircraft so that the aircraft always carries the maximum allowable. This repair operator was selected to increase the exploration capacity of the GA. The repair algorithm is as follows:

Algorithm 8 GA offspring gene repair operator

if $\sum_{i=1}^{M} w_i x_{ij} < c_j$ for any j **then**
For each j where (6.2) is violated
while $\sum_{i=1}^{M} w_i x_{ij} < c_j$ **do**

- randomly select an element i from the gene structure for aircraft j
- Set $x_{ij} = x_{ij} + 1$

end while
else
if $\sum_{i=1}^{M} w_i x_{ij} > c_j$ for any j **then**
For each j where (6.2) is violated
while $\sum_{i=1}^{M} w_i x_{ij} > c_j$ **do**

- randomly select an element i from the gene structure for aircraft j
- Set $x_{ij} = x_{ij} - 1$

end while
end if
end if

This operator is easily implemented and provides further exploration of the design space because of its random nature.

6.4.2 Solution of the DWTA.

As stated, Method 2 integrates an approximate dynamic programming routine to reduce the space investigated by the GA. Instead of using the design structure presented in Section 6.4.1 it is updated to the following.

$$D(P) = (R_1, R_2, \ldots, R_M) \tag{6.17}$$

This design point is then fed to the ADP routine which determines a_{tij} for all $t \in \mathcal{T}$, $i = 1, \ldots, m$, and $j = 1, \ldots, n$. This is represented in Figure 11

6.5 Numerical Results and Discussion

Two sets of experiments were performed to determine the efficacy of each method. The first set of experiments explores the case where $c_1 = 6$ and $c_2 = 2$, and there are five targets following $Y_t = (2, 1, 1, 1)^T$ and $V = (100, 150, 200, 300)$. As in [76], the probabilities which define state transitions for both sets of experiments are defined in Table 21.

Table 21. Updated conditional transition probabilities

	Single Shot p_{ry} (all target types remain)				No SAMs Remaining		No Radars Remaining			No SAM or Radar
Weapon Type	SAM 1	SAM 2	Radar	C^2	Radar	C^2	SAM 1	SAM 2	C^2	C^2
1	0.47	0.51	0.6	0.59	0.67	0.69	0.75	0.65	0.76	0.76
2	0.53	0.68	0.58	0.54	0.65	0.83	0.87	0.7	0.84	0.92
3	0.48	0.56	0.47	0.51	0.58	0.89	0.86	0.94	0.86	0.91
4	0.55	0.58	0.48	0.56	0.7	0.93	0.68	0.64	0.88	0.94
5	0.47	0.65	0.45	0.62	0.83	0.78	0.54	0.71	0.82	0.83
6	0.56	0.48	0.51	0.47	0.74	0.77	0.55	0.78	0.84	0.9
7	0.64	0.51	0.56	0.48	0.65	0.95	0.7	0.68	0.94	0.95

After several iterations, a population size of 50 was selected as a reasonable size to begin exploration of the design space. For each method, the same initial population was used, and, as appropriate, common random numbers were used to reduce experimental noise. In addition, because of the convergence properties demonstrated, 50 generations were used. A representative example of the experimental results are presented in Figure 13.

In this instance, the baseline GA with randomly generated sequencing outperformed the integrated ADP GA method, though solution quality may be of practical insignificance. For the baseline GA, the solution is $\mathbf{x} = (x_{\cdot 1} = (2, 1, 3, 0), x_{\cdot 2} = (0, 0, 2))$ and the weapons would be fired (myopically) over two stages. For the

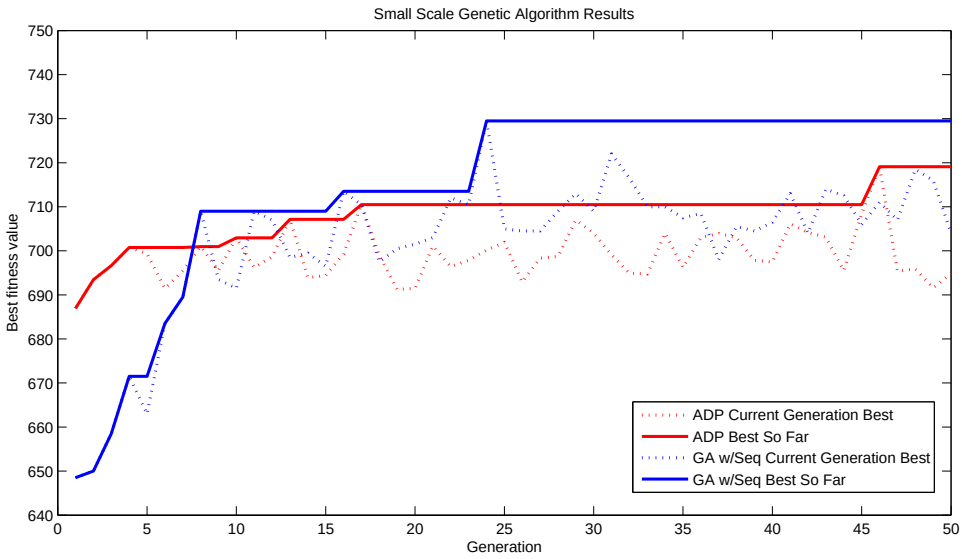


Figure 13. Plot of small scale genetic algorithm results

integrated ADP method the solution is $\mathbf{x} = (x_{\cdot 1} = (0, 4, 0, 0)$ and $x_{\cdot 2} = (2, 0, 0))$. In both cases, the mix of weapons was converged to rather quickly (approximately 10 generations), while the sequencing or allocations strategy continued evolving. As a note, from an acquisition perspective, because the development, maintenance, and sustainment costs associated with numerous high-value weapon types, the integrated ADP method may have resulted in a more desirable solution.

The second set of experiments were on a slightly larger problem where $c_1 = 8, c_2 = 2, Y_t = (2, 2, 2, 1)^T$, and $V = (100, 150, 200, 300)$. The results for the larger problem are shown in Figure 14.

For the second experiment, Figure 14 shows the marked improvement in solution quality using the integrated ADP method. Consistent with the results found in [76], there is an approximately 15.5% improvement over the random sequencing with myopic allocation. One explanation in the difference in results is because, as problem size increases, there is a greater benefit of allocating weapons while considering the impact those allocations may have on the future of the system. The solutions for the

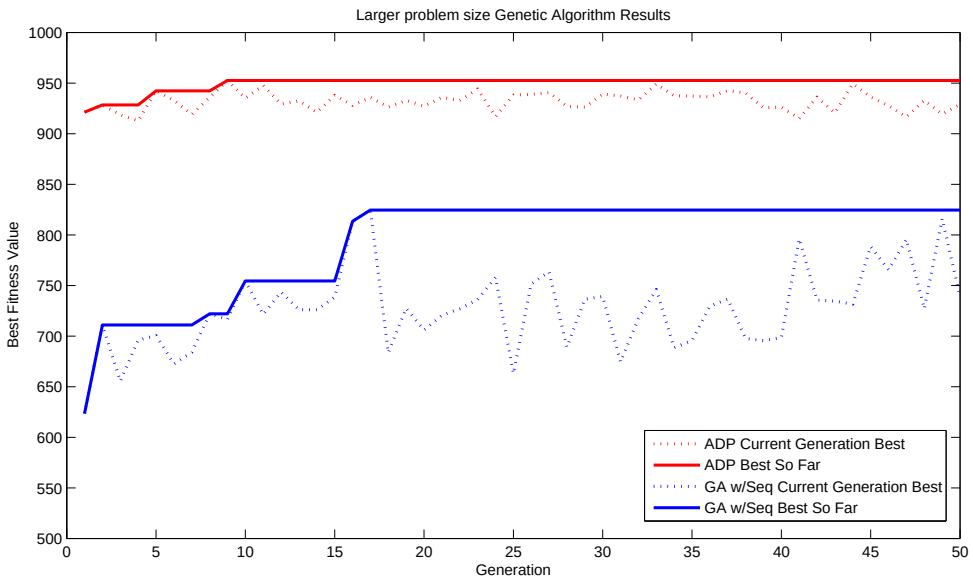


Figure 14. Plot of small scale genetic algorithm results

baseline and ADP methods are $\mathbf{x} = (x_{\cdot 1} = (2, 5, 1, 0),\ x_{\cdot 2} = (0, 1, 1))$ and $\mathbf{x} = (x_{\cdot 1} = (2, 5, 0, 1),\ x_{\cdot 2} = (1, 0, 1))$, respectively. The fact that both methods converged to very similar solutions provides some validation of the proposed solution framework. It also further emphasizes the impact that the system dynamics will have on the solution. Additionally, looking at the set of solutions presented, weapon type two appears to be a dominant weapon that would be of interest to those making critical acquisition decisions.

6.6 Conclusions

This research presents an embedded optimization problem in which the solution of a WTA problem is used to determine the utility needed to solve a multidimensional knapsack problem. Two methods were presented that are shown to converge to quality solutions using different allocation determinations. For larger problem sizes, the integrated ADP method outperforms the baseline method with random sequencing

and myopic allocation. The quality in solution, however, comes at a price. Because each gene represents a unique optimization problem, the ADP method must load problem data into memory prior to executing a solution. As problem size increases, this may be computationally impractical when compared to the random sequencing method. Since both methods converged to a similar solution for the multidimensional knapsack problem, it may be more effective to do a quick GA search of the space using the baseline method and follow it up using ADP to determine a better employment strategy. This may, however, be mitigated through the use of better computing languages, higher powered computers, distributed computing. Additionally, numerous other areas will be explored in this ongoing research area. First, this formulation assumes known weapons effects, when that may not necessarily be the case. Future research will consider Bayesian updates of the kill probabilities as a feedback from the simulation outputs. Additionally, because they have been shown to be effective, hybrid heuristics may be explored to improve solution quality in fewer generations. As alluded to, analysts may be interested in only exploring a few weapon types further, so constraints may be added to the knapsack problem that reduce the number of weapon types allowed in any single gene structure. Similarly, if certain weapon types are able to be used on either aircraft, but some weapon types are only allowed on a specific aircraft, complexity increases. Another area would use heuristics for the static WTA problem solved that will consider the impact of allocations on future events. This may help increase solution quality for the much faster baseline method. Lastly, the ultimate purpose of this is to integrate it within a high-level combat simulation in lieu of the simple Monte Carlo simulation presented above. This will provide analysts with a means for effectively determining which weapons concepts to explore further, how to appropriately fit a set of aircraft with these weapon types, and how to effectively employ them within a given scenario.

VII. Conclusions and Recomendations

Several conclusions can be highlighted based on DP extensions and computational results. This chapter reviews the research, provides concluding insights about the results, and identifies topics for future research efforts.

7.1 Summary of Effort

Throughout this effort, several significant and original contributions are made to the field of operations research by developing new models for investigation and identifying novel solution techniques and performing computational studies. First, an efficient solution methodology is presented that determines optimal weapons allocation for a two-stage DWTA problem instance. This is the first provably optimal algorithm for this problem instance. Next, the two-stage problem is extended and considers the dependency across stages when determining allocation policies which demonstrates improvement over existing methods and effective scalability for large problems. In addition, this dissertation formulates and solves a previously undefined instance of the DWTA problem that incorporates dynamic probabilities of kill using problem structure to develop effective solution strategies. To address this problem, a rigorous and novel approximate dynamic programming method is developed which reduces the size of the decision space to a more computationally tractable size. Several distributions were investigated which use the problem structure to reinforce the selection quality decisions. Finally, an embedded optimization problem which seeks to optimize an aircraft weaponeering policy is developed. This optimization problem defines the utility of a weapon through the solution to a weapon-target assignment problem. These utilities are then used to solve a constrained multi-dimensional knapsack problem that represents placing weapons on a set of aircraft. A GA is used as

the solution framework, and two techniques that integrate the sequential allocation of weapons into the gene structure are compared and contrasted. Through the development of the GA, this dissertation effectively determines locally optimal weaponeering policies. In addition, this research develops a defensible methodology for real-time allocation strategies within simulation applications for current practitioners.

7.2 Conclusion

Results for this research demonstrate the contribution of the effort. In each case tested, high quality solutions are generated in much less computation time, comparatively. For the two-stage problem, the algorithms ability to determine optimal solutions is proven through several theorems. Further, the computational complexity of the method is shown to provide solutions in a much more efficient manner. Next, computational results for the two-stage extension demonstrate the effectiveness of the adaptive dynamic programming methodology in obtaining near optimal solutions for various problem instances in much less computation times than what currently exists in the literature. Additionally, results show a substantial improvement in solution quality in less computation time than other techniques have demonstrated as problem size increases. Because of the combinatorial nature of the weapon target assignment, determining an exact solution using dynamic programming is computationally intractable. Further, current literature does not provide methods to appropriately address the vast size of the decision space for any given state. The solution methodology presented in this research greatly reduces the size of the decision space necessary for investigation, and exploits the special structure of the problem to maintain solution quality in an efficient manner. Finally, by integrating the sequential allocation policies into a GA, two options are available which trade off computation time for solution quality when determining optimal weapons mix. Results show that random genera-

tion of the sequence with a myopic allocation strategy is fast, but does not give the solution quality provided by determining near optimal sequential allocation policies using ADP. Overall, this research presents a defensible approach that addresses gaps in the literature and novel approaches for the solution of the motivating problems. In each case, numerous tests are run and the results presented. As with many research efforts, as many questions get answered, new questions arise.

7.3 Future work

With each of the presented areas of research comes a stream of potential future research. Extensions may be investigated for each of the problem types, along with the increase in complexity through the alteration of assumptions. For each effort, an associated discussion of future research is presented.

7.3.1 Shoot-look-shoot.

The two-stage DWTA problem has many identifiable extensions. First, the model can be extended to include the impact of cost on the approximation scheme as well as the effect sensors may have in the first stage, second stage, or across both stages. Additionally, as weapons for this effort are currently homogeneous within a stage, a natural extension will investigate non-homogeneous weapons in and across stages. Further, because the subgradients represent the marginal increase in reserving a weapon for future stages, the algorithm may be very effective in instances where there are more than two stages. Therefore, additional research may extend this to multiple stages. Finally, the presented method starts with all weapons initially allocated in stage one. This research may be extended to explore the initial allocation of weapons in the second stage, or some other initial allocation policy.

7.3.2 Cooperative DWTA Problem.

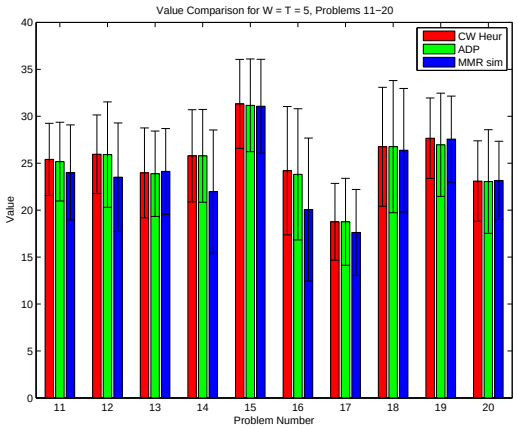
Because this is a novel formulation, there is an extensive list of future work. First, different approximate dynamic programming techniques should be investigated to address the dimensionality of the decision space. Though it was slower computationally, implementing a multi-step look ahead solution within the myopic framework may result in better solution quality because it is an exact method in the sense that it iterates over the full state and decision spaces. Additionally, a reduced decision space could be coupled with state reduction techniques such as aggregation to further reduce computation time while maintaining solution quality. Finally, investigating roll-out algorithms which take into account the future impact of current decisions may be implementable within a simulation framework to quickly determine optimal policies for problems of a larger size.

7.3.3 Embedded Optimization Framework.

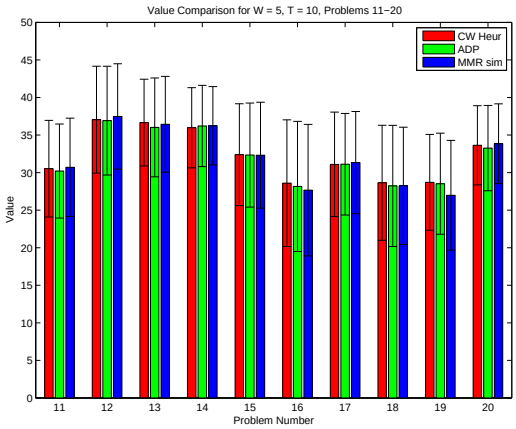
Finally, the embedded optimization framework is in its infancy and much is left to be accomplished. First, the present formulation assumes known weapons effects, though because future weapons concepts are being investigated, weapons effects are likely unknown. Future research will consider Bayesian updates of the kill probabilities as a feedback from the simulation outputs. Additionally, because they have been shown to be effective, hybrid heuristics may be explored to further improve solution quality. As alluded to, analysts may be interested in only exploring a few weapon types further, so constraints may be added to the knapsack problem that reduce the number of weapon types allowed in any single gene structure. Similarly, aircraft-specific weapons may be investigated as an additional constraint in the model. This would likely increase the complexity of the model and may impact the effectiveness of the developed solution methodology. Another area would use heuristics for the static

WTA problem solved that consider the impact of allocations on future events. This may help increase solution quality for the much faster baseline method. The ultimate purpose of this research effort is to integrate it within a high-level combat simulation in lieu of the simple Monte Carlo simulation. This will provide analysts with a means for effectively determining which weapons concepts to explore further, how to appropriately fit a set of aircraft with these weapon types, and how to effectively employ them within a given scenario. Lastly, making use of distributed computing as well as high-powered computing resources should be investigated to assist with real-time decision making.

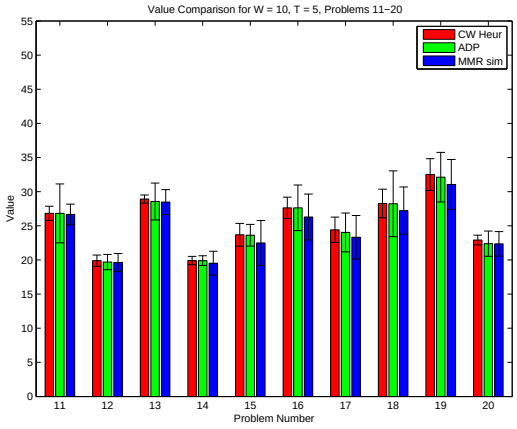
Appendix A. Data Tables and additional figures



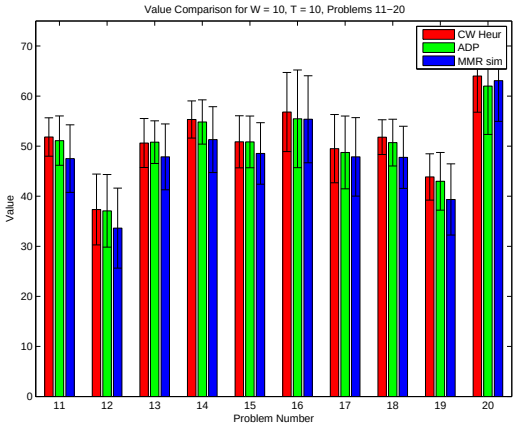
(a) W = 5, T = 5



(b) W = 5, T = 10

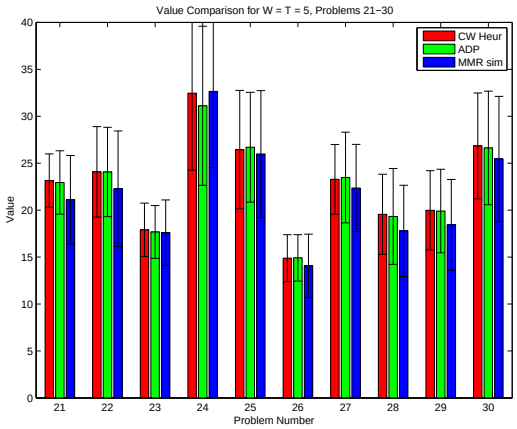


(c) W = 10, T = 5

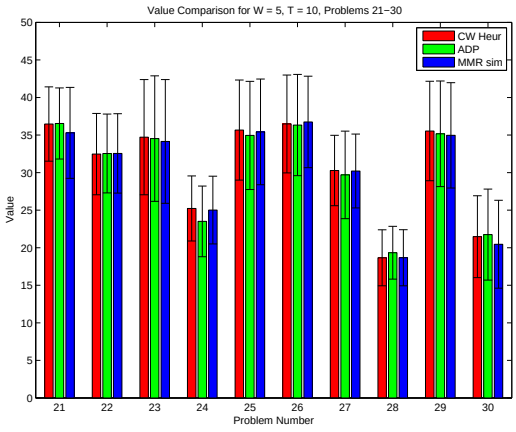


(d) W = 10, T = 10

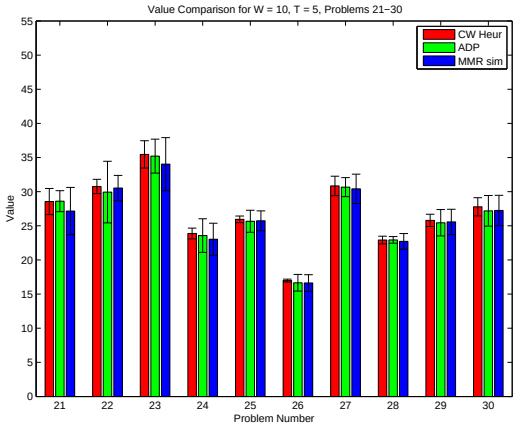
Figure 15. Results for small sized experiments at varying W & T



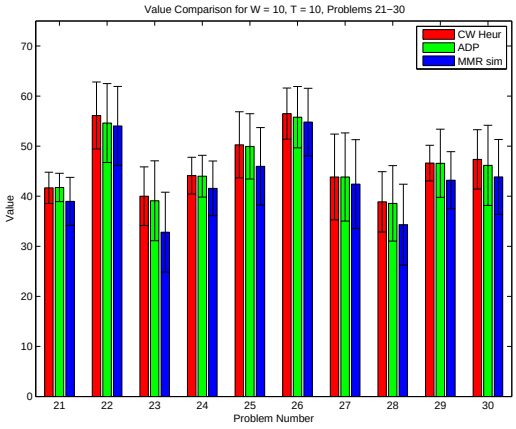
(a) W = 5, T = 5



(b) W = 5, T = 10

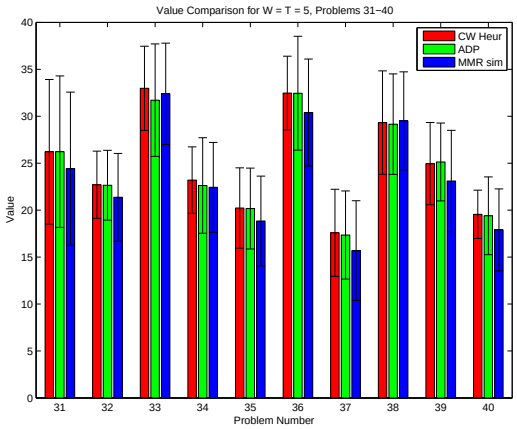


(c) W = 10, T = 5

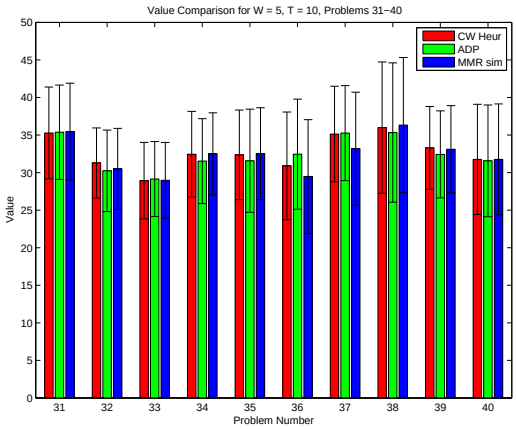


(d) W = 10, T = 10

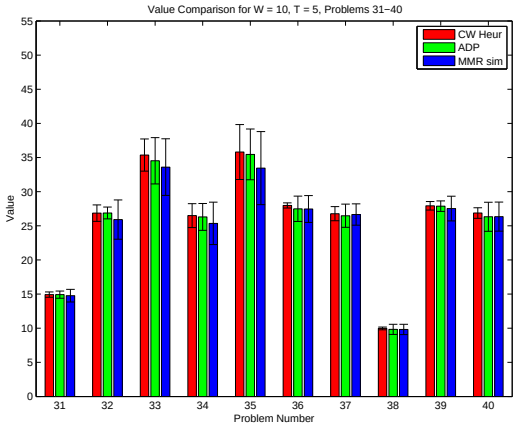
Figure 16. Results for small sized experiments at varying W & T



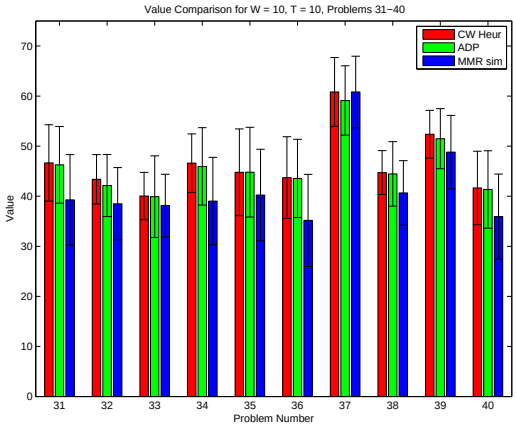
(a) W = 5, T = 5



(b) W = 5, T = 10

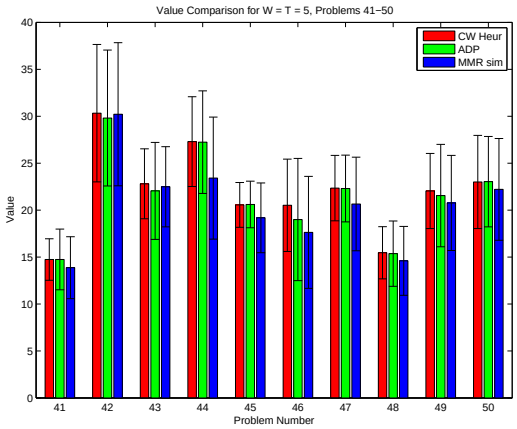


(c) W = 10, T = 5

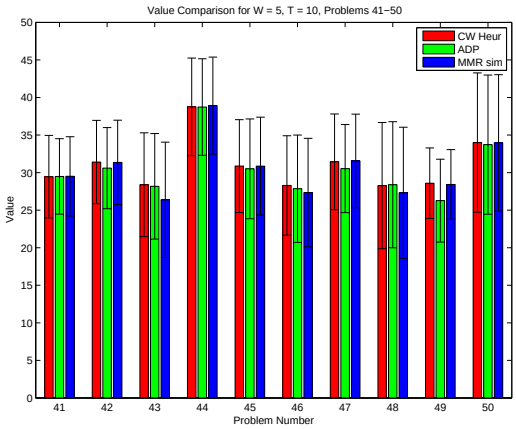


(d) W = 10, T = 10

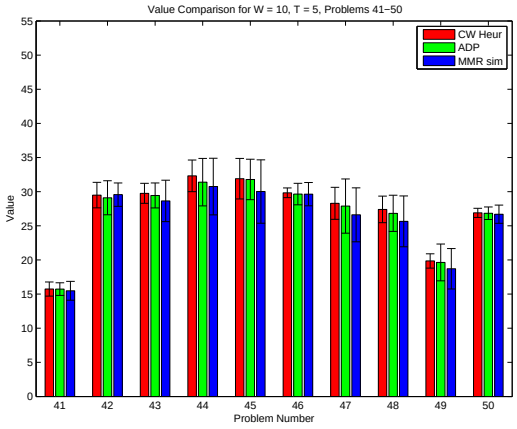
Figure 17. Results for small sized experiments at varying W & T



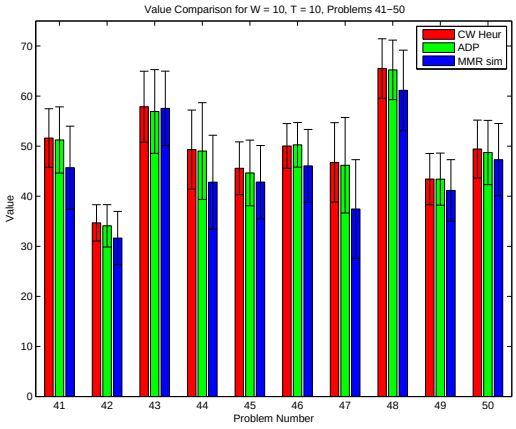
(a) W = 5, T = 5



(b) W = 5, T = 10

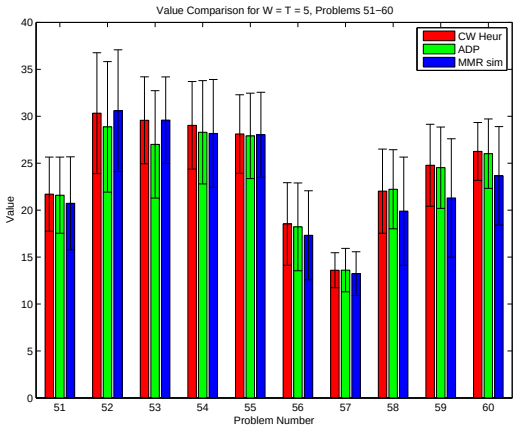


(c) W = 10, T = 5

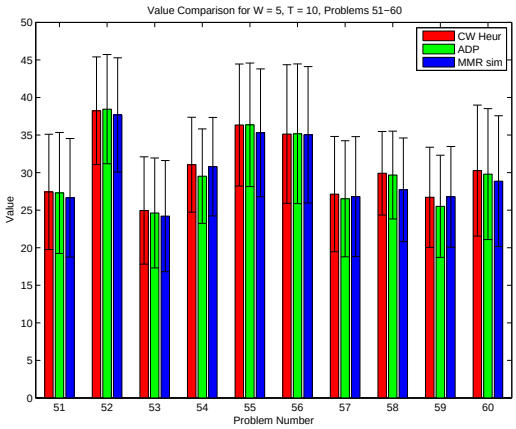


(d) W = 10, T = 10

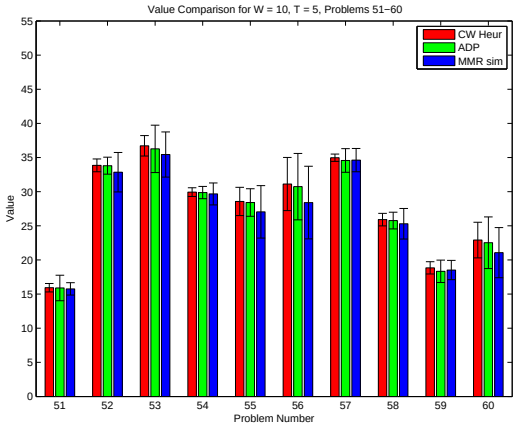
Figure 18. Results for small sized experiments at varying W & T



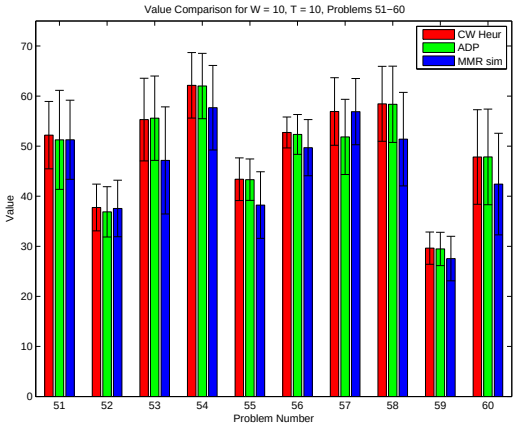
(a) W = 5, T = 5



(b) W = 5, T = 10

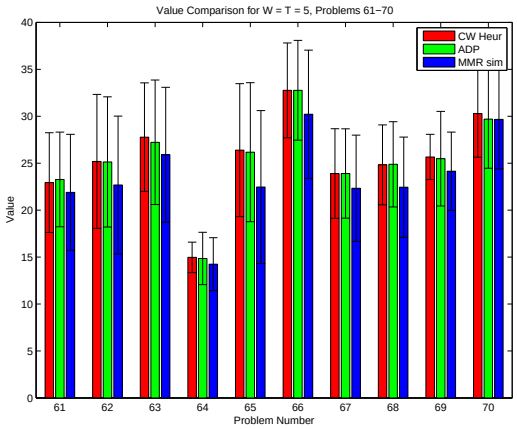


(c) W = 10, T = 5

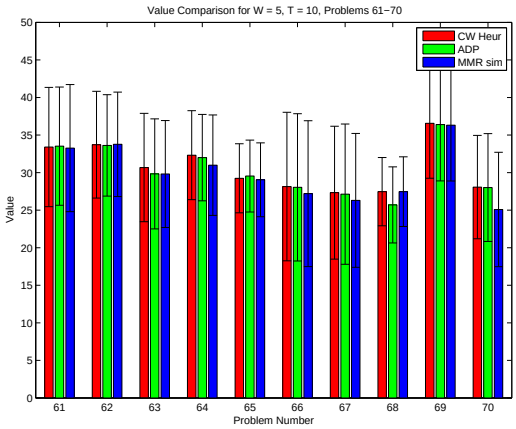


(d) W = 10, T = 10

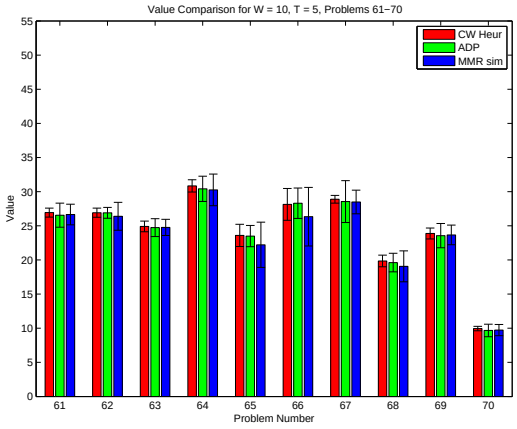
Figure 19. Results for small sized experiments at varying W & T



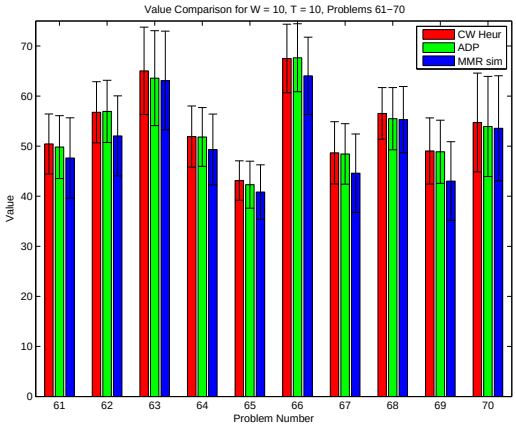
(a) W = 5, T = 5



(b) W = 5, T = 10

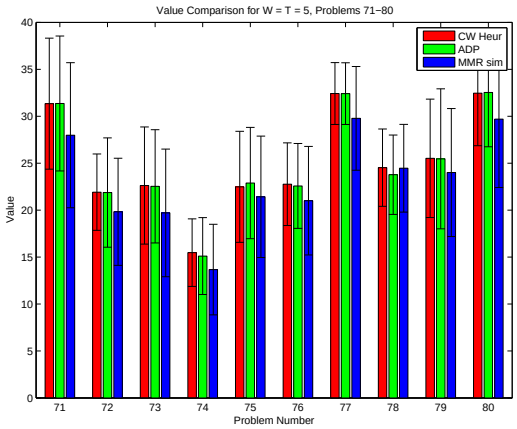


(c) W = 10, T = 5

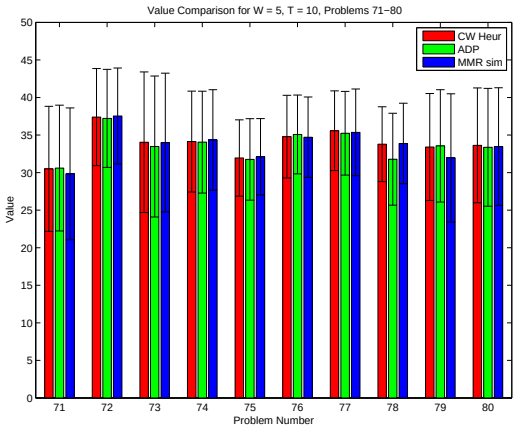


(d) W = 10, T = 10

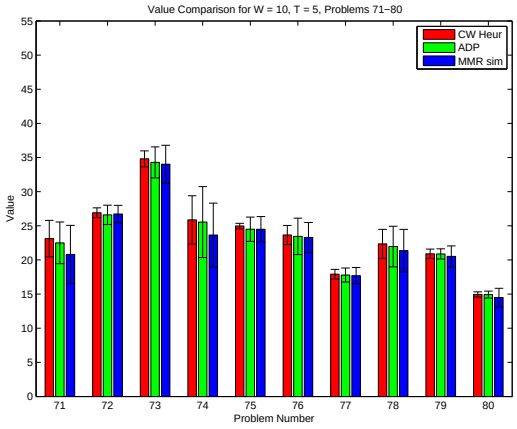
Figure 20. Results for small sized experiments at varying W & T



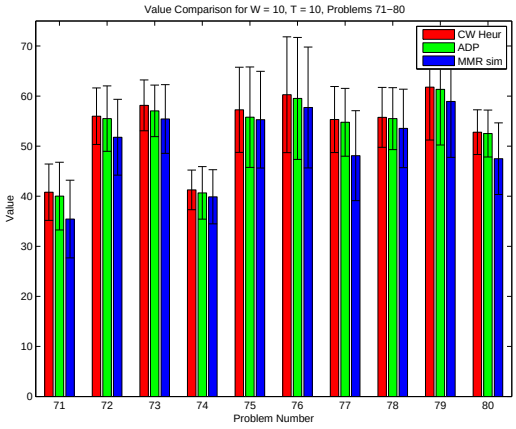
(a) W = 5, T = 5



(b) W = 5, T = 10

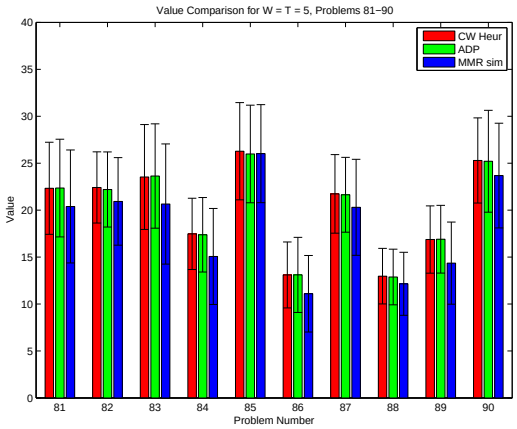


(c) W = 10, T = 5

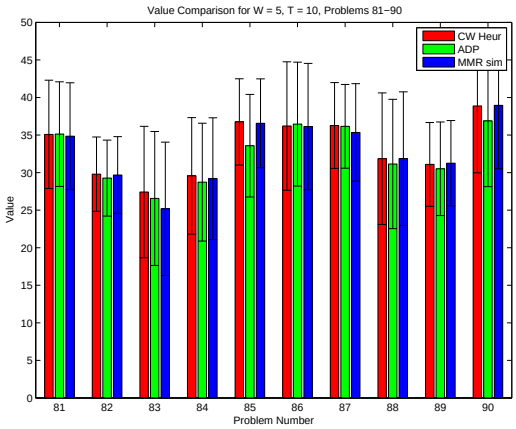


(d) W = 10, T = 10

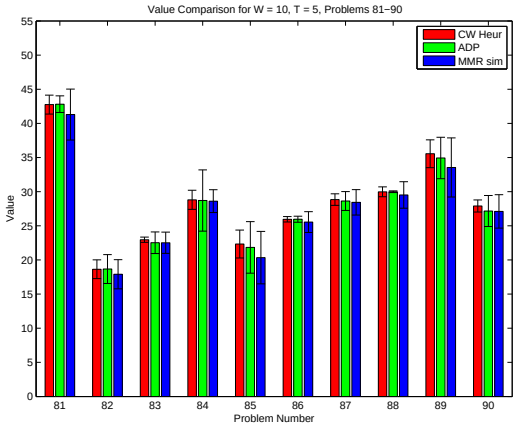
Figure 21. Results for small sized experiments at varying W & T



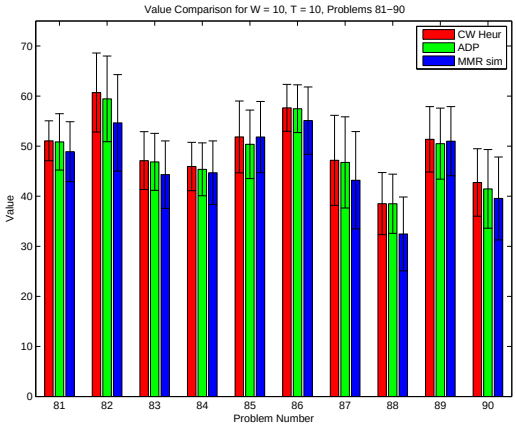
(a) W = 5, T = 5



(b) W = 5, T = 10

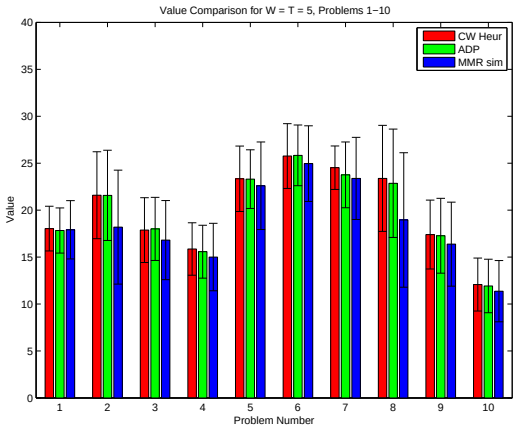


(c) W = 10, T = 5

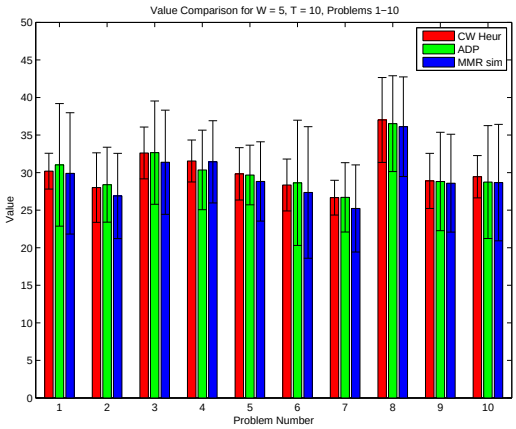


(d) W = 10, T = 10

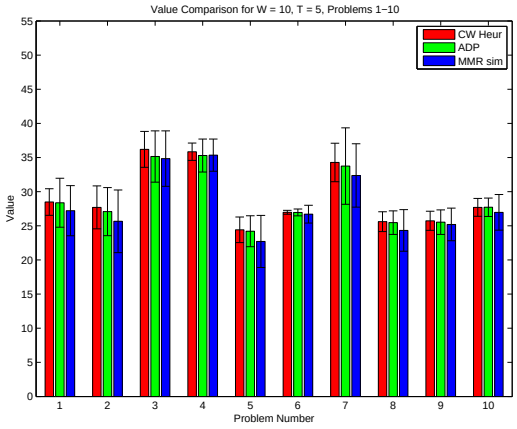
Figure 22. Results for small sized experiments at varying W & T



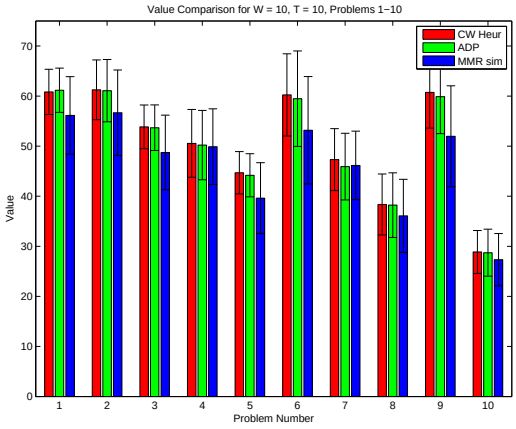
(a) W = 5, T = 5



(b) W = 5, T = 10

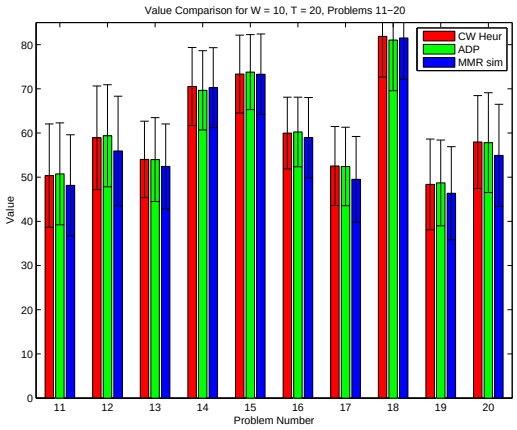


(c) W = 10, T = 5

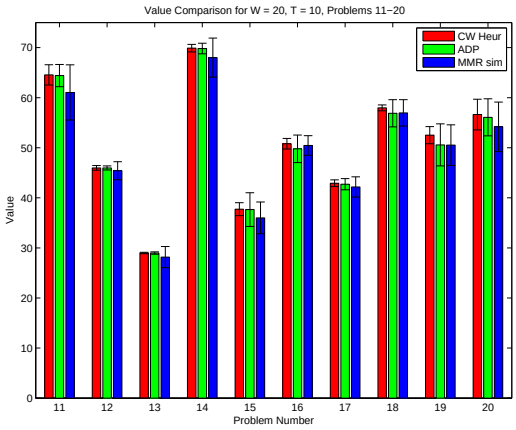


(d) W = 10, T = 10

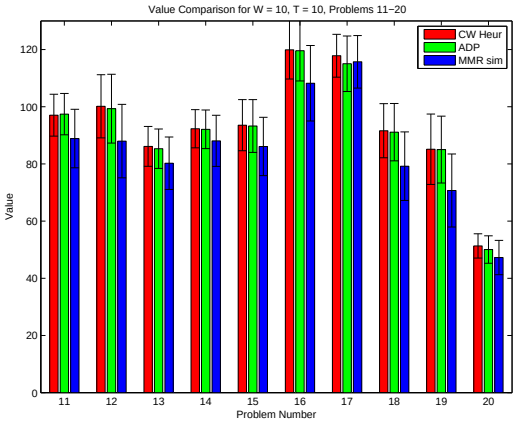
Figure 23. Results for small sized experiments at varying W & T



(a) W = 10, T = 20

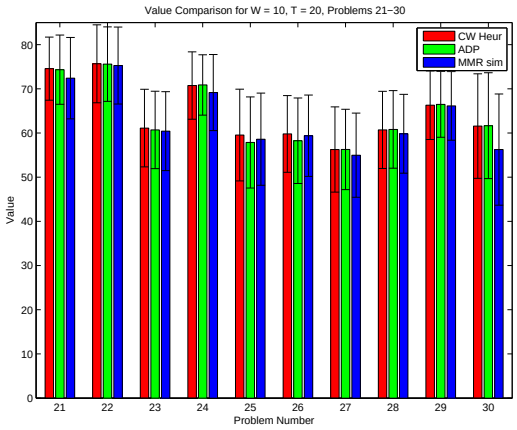


(b) W = 20, T = 10

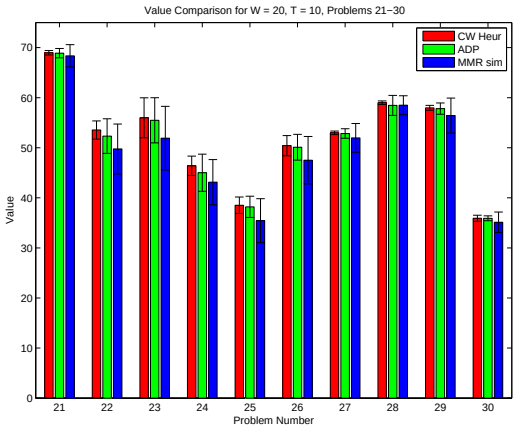


(c) W = 20, T = 20

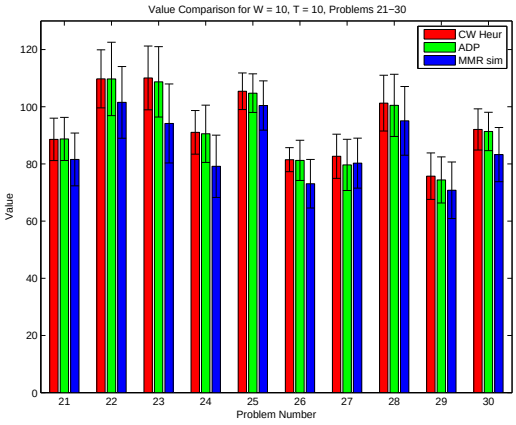
Figure 24. Results for medium sized experiments at varying W & T



(a) W = 10, T = 20

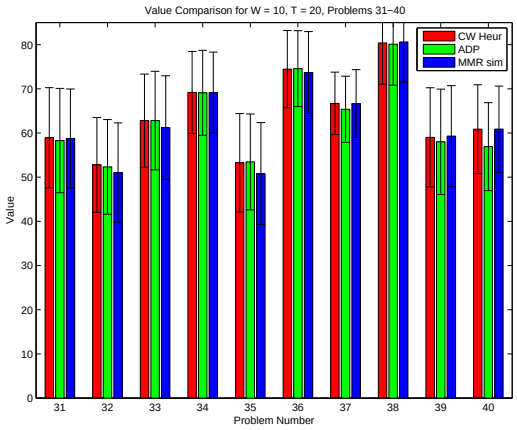


(b) W = 20, T = 10

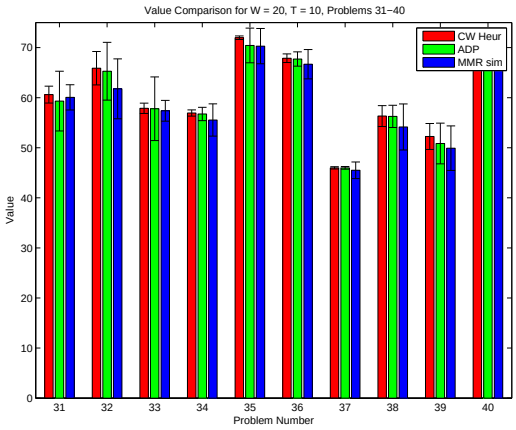


(c) W = 20, T = 20

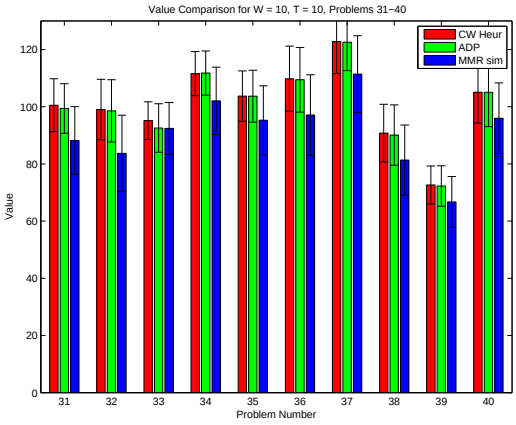
Figure 25. Results for medium sized experiments at varying W & T



(a) W = 10, T = 20

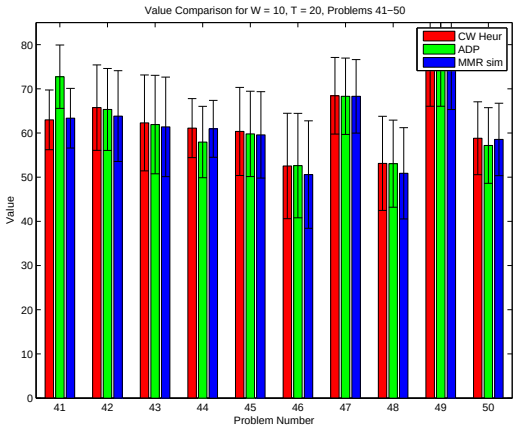


(b) W = 20, T = 10

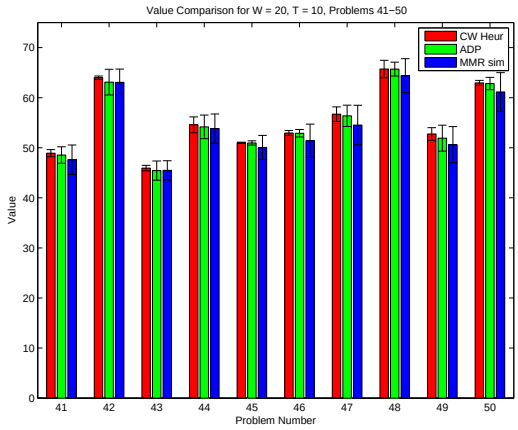


(c) W = 20, T = 20

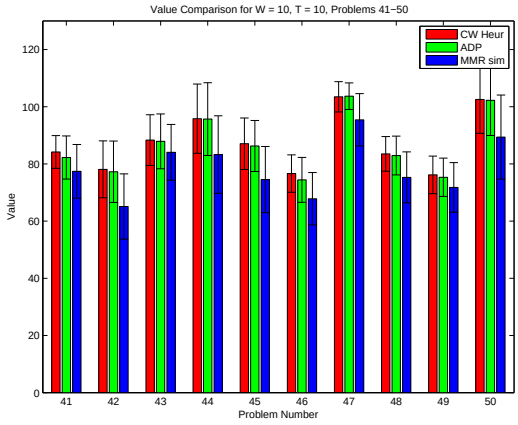
Figure 26. Results for medium sized experiments at varying W & T



(a) W = 10, T = 20

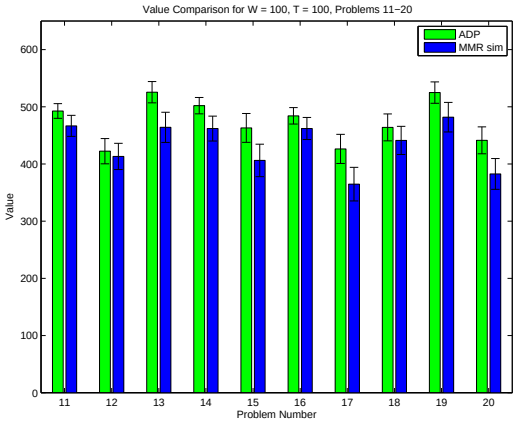


(b) W = 20, T = 10

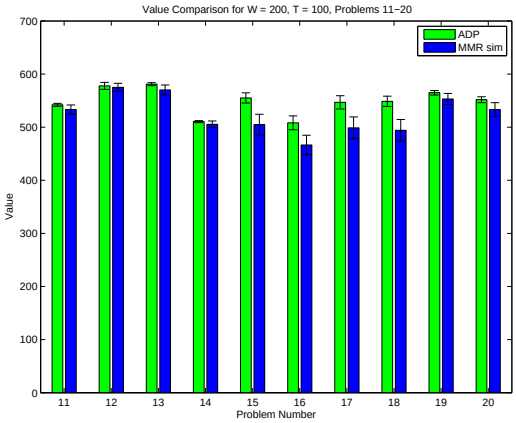


(c) W = 20, T = 20

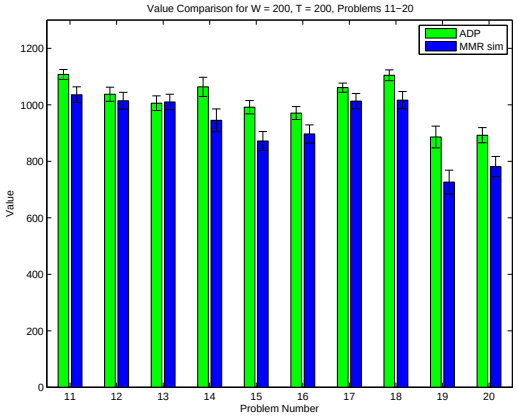
Figure 27. Results for medium sized experiments at varying W & T



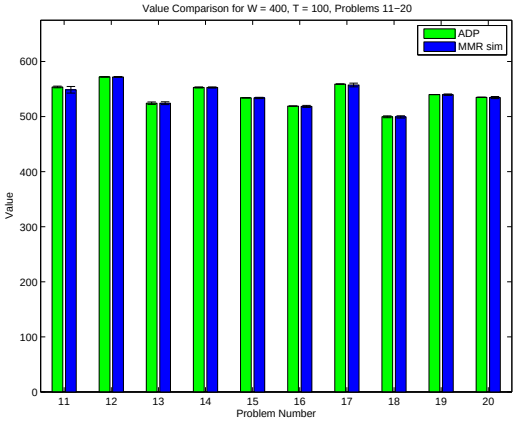
(a) W = 100, T = 100



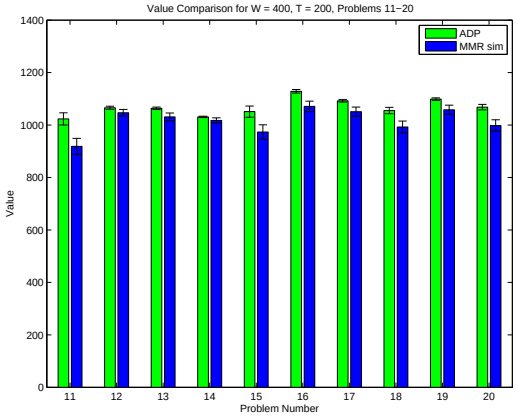
(b) W = 200, T = 100



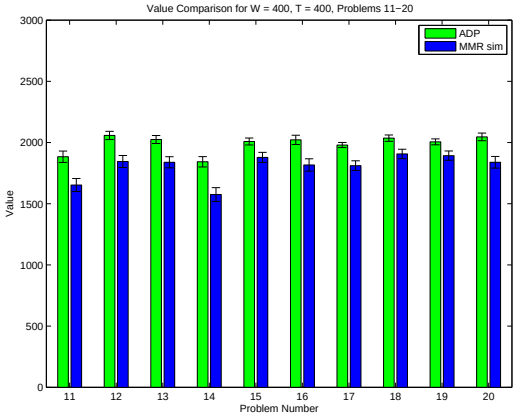
(c) W = 200, T = 200



(d) W = 400, T = 100

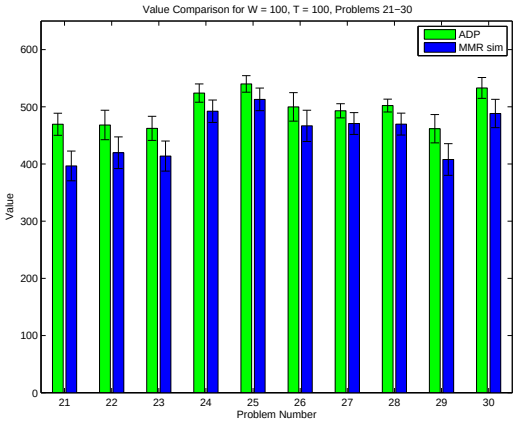


(e) W = 400, T = 200

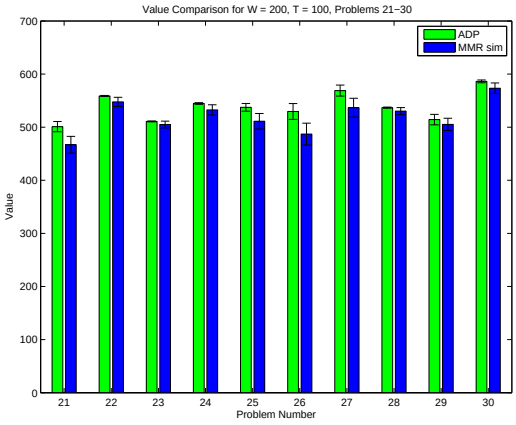


(f) W = 400, T = 400

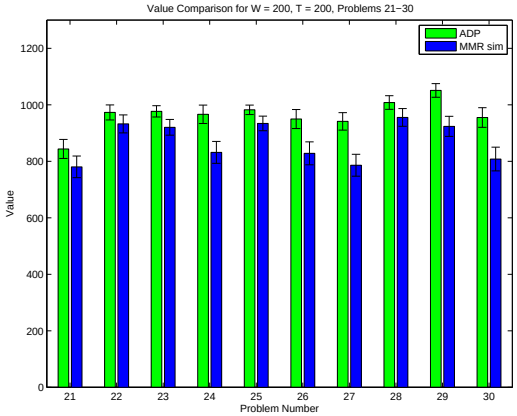
Figure 28. Results for first 10 large scale experiments at varying W & T



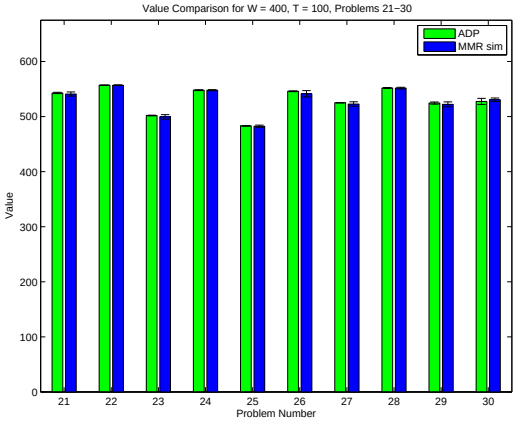
(a) W = 100, T = 100



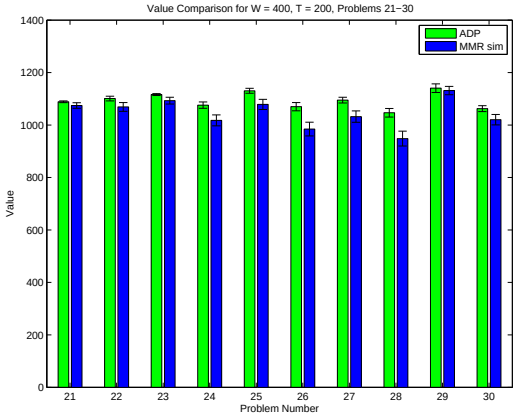
(b) W = 200, T = 100



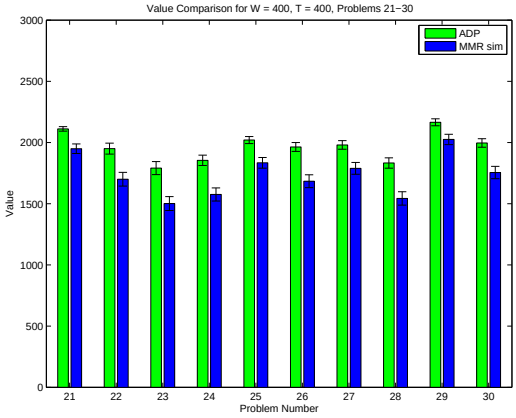
(c) W = 200, T = 200



(d) W = 400, T = 100

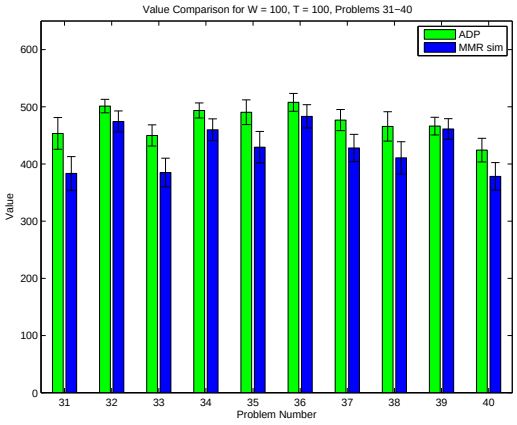


(e) W = 400, T = 200

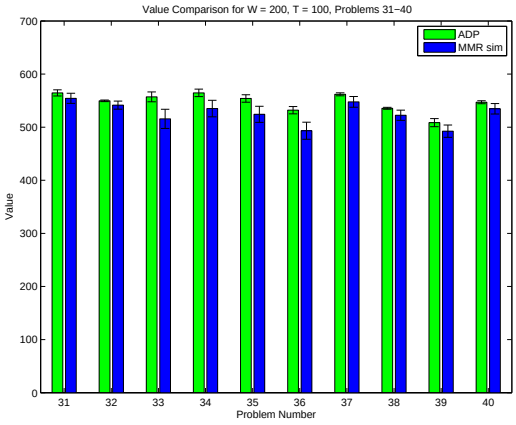


(f) W = 400, T = 400

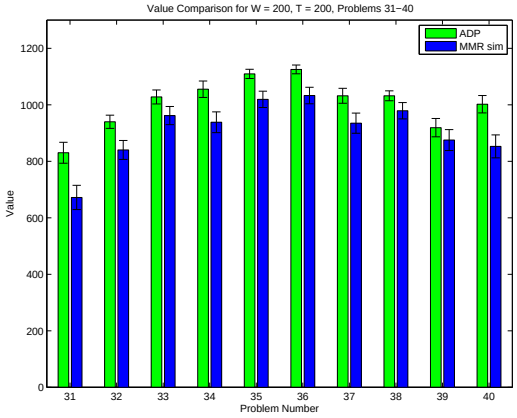
Figure 29. Results for first 10 large scale experiments at varying W & T



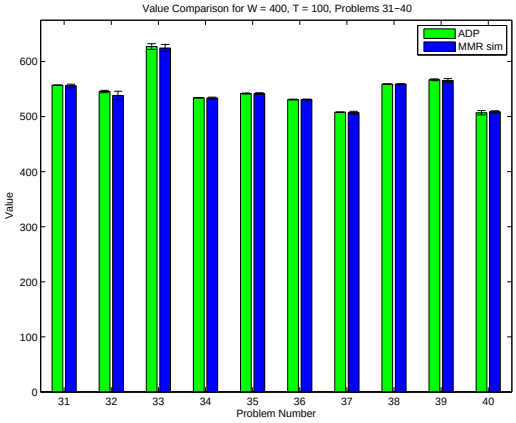
(a) W = 100, T = 100



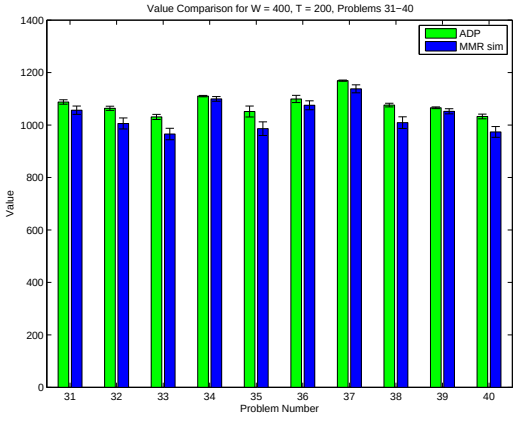
(b) W = 200, T = 100



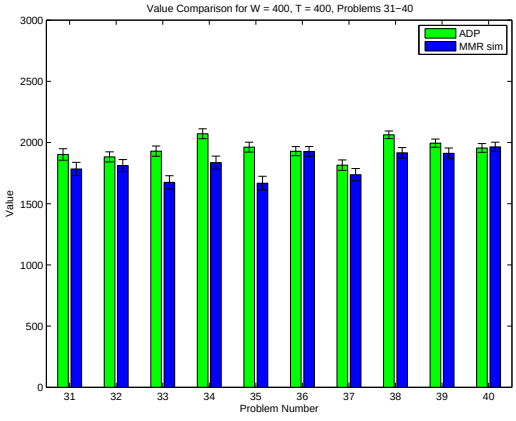
(c) W = 200, T = 200



(d) W = 400, T = 100

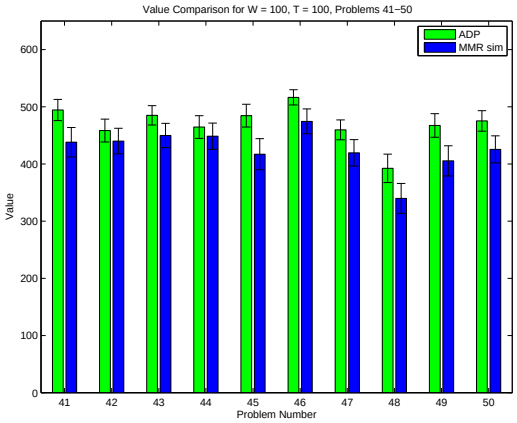


(e) W = 400, T = 200

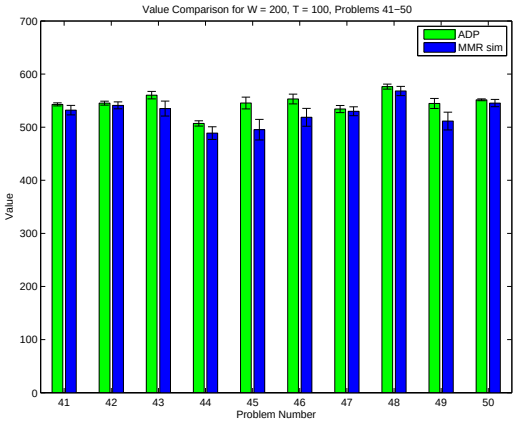


(f) W = 400, T = 400

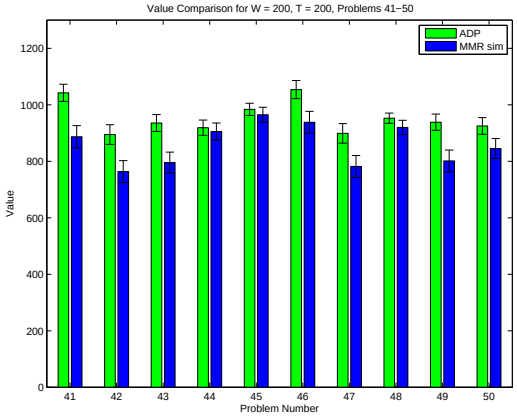
Figure 30. Results for first 10 large scale experiments at varying W & T



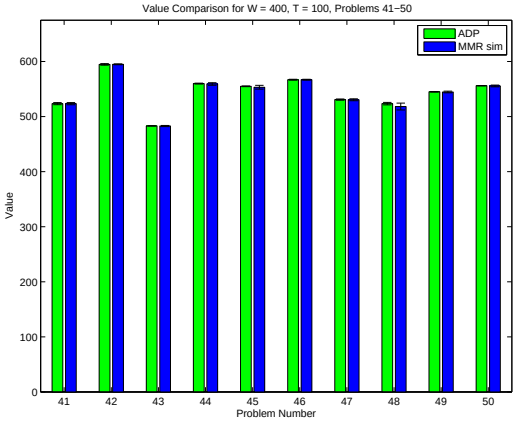
(a) W = 100, T = 100



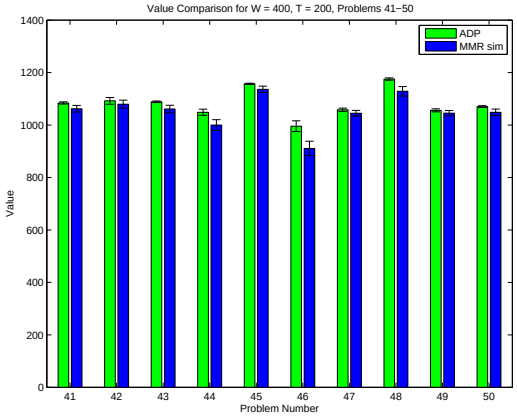
(b) W = 200, T = 100



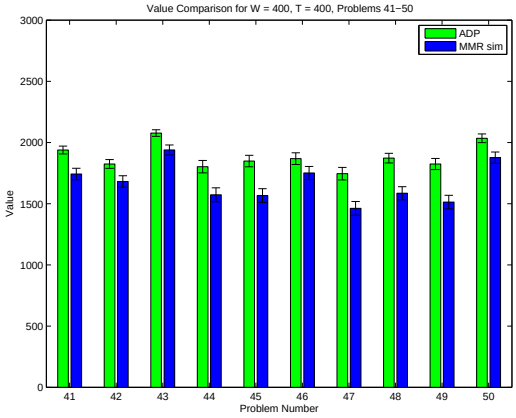
(c) W = 200, T = 200



(d) W = 400, T = 100



(e) W = 400, T = 200



(f) W = 400, T = 400

Figure 31. Results for first 10 large scale experiments at varying W & T

Bibliography

[1] D. Adelman. Overview of approximate dynamic programming using math programming. Industrial and Systems Engineering Research Conference Tutorial Session, 2013.

[2] D.K. Ahner. Planning and control of unmanned aerial vehicles in a stochastic environment. *Doctor of Philosophy dissertation, Boston University*, 2005.

[3] D.K. Ahner. Real-time planning and control of army uavs under uncertainty. *AIAA Journal of Aerospace Computing, Information and Communication*, (4):798–815, 2007.

[4] D.K. Ahner and C.R. Parson. Optimal multi-stage allocation of weapons to targets using adaptive dynamic programming. *Optim Lett*, 2014.

[5] Kumar A. Jha K.C. Ahuja, R.K. and J.B. Orlin. Exact and heuristic algorithms for the weapon-target assignment problem. *Operations Research*, 55(6):1136–1146, 2007.

[6] Akin E. Alatas, B. and A.B. Ozer. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals*, 40(4):1715–1734, 2009.

[7] Marden J.R. Arslan, G. and J.S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[8] Y. Aviv and M. Kress. Evaluating the effectiveness of shoot-look-shoot tactics in the presence of incomplete damage information. *Military Operations Research*, 3(1):79–89, 1997.

[9] S.N. Bashi-Azghadi and R. Kerachian. Locating monitoring wells in groundwater systems using embedded optimization and simulation models. *Science of the Total Environment*, 408(10):2189–2198, 2010.

[10] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[11] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2007.

[12] D.P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific, 2012.

[13] D.P. Bertsekas. *Dynamic programming and optimal control*, volume 2. Athena Scientific, 2012.

[14] D.P. Bertsekas and D.A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[15] D.P. Bertsekas and J. Tsitsiklis. *Neuro Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[16] Homer M.L. Logan D.A. Patek S.D. Bertsekas, D.P. and N.R. Sandell. Missile defense and interceptor allocation by neuro-dynamic programming. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(1):42–51, 2000.

[17] Jerome Bracken, James E Falk, and Frederic A Miercort. A strategic weapons exchange allocation model. *Operations Research*, 25(6):968–976, 1977.

[18] Gerald Brown, Matthew Carlyle, Douglas Diehl, Jeffrey Kline, and Kevin Wood. A two-sided optimization for theater ballistic missile defense. *Operations research*, 53(5):745–763, 2005.

[19] QY Cao and ZB He. A genetic algorithm of solving wta problem. *Control Theory and Applications*, 18(1):76279, 2001.

[20] D.A. Castanon. Advanced weapon-target assignment algorithm quarterly report. *Tr-337, ALPHA TECH Inc., Burlington, Massachusetts*, 1987.

[21] D.A. Castanon. Approximate dynamic programming for sensor management. In *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, volume 2, pages 1202–1207. IEEE, 1997.

[22] D.A. Castanon and J.M. Wohletz. Model predictive control for dynamic unreliable resource allocation. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 4, pages 3754–3759. IEEE, 2002.

[23] D.A. Castanon and J.M. Wohletz. Model predictive control for stochastic resource allocation. *Automatic Control, IEEE Transactions on*, 54(8):1739–1750, 2009.

[24] James R.M. Chang, S.C. and J.J. Shaw. Assignment algorithm for kinetic energy weapons in boost phase defence. In *Decision and Control, 1987. 26th IEEE Conference on*, volume 26, pages 1678–1683. IEEE, 1987.

[25] Jie Chen, Bin Xin, ZhiHong Peng, LiHua Dou, and Juan Zhang. Evolutionary decision-makings for the dynamic weapon-target assignment problem. *Science in China Series F: Information Sciences*, 52(11):2006–2018, 2009.

[26] Lina Chen, ChuanJun Ren, and Su Deng. An efficient approximation for weapon-target assignment. In *Computing, Communication, Control, and Management, 2008. CCCM'08. ISECS International Colloquium on*, volume 1, pages 764–767. IEEE, 2008.

[27] P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[28] R.H. Day. Allocating weapons to target complexes by means of nonlinear programming. *Operations Research*, 14(6):992–1013, 1966.

[29] E.V. Denardo. *Dynamic programming: models and applications*. Dover Publications, 1982.

[30] Ellison R.E. denBroeder, Jr. G.G. and L. Emerling. On optimum target assignments. *Operations Research*, 7(3):322–326, 1959.

[31] A.R. Eckler and S.A. Burr. Mathematical models of target coverage and missile allocation. Technical report, DTIC Document, 1972.

[32] Salah E Elmaghraby. Resource allocation via dynamic programming in activity networks. *European Journal of Operational Research*, 64(2):199–215, 1993.

[33] Tiao-ping Fu, Yu-shu Liu, and Jian-hua Chen. Improved genetic and ant colony optimization algorithm for regional air defense wta problem. In *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, volume 1, pages 226–229. IEEE, 2006.

[34] Shang Gao and JY Yang. Solving weapon-target assignment problem by particle swarm optimization algorithm. *Systems Engineering and Electronics*, 27(7):1250–1252, 2005.

[35] K. Glazebrook and A. Washburn. Shoot-look-shoot: A review and extension. *Operations Research*, 52(3):454–463, 2004.

[36] F. Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[37] G.A. Godfrey and W.B. Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, 2001.

[38] G.A. Godfrey and W.B. Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, 2001.

[39] G.A. Godfrey and W.B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science*, 36(1):21–39, 2002.

[40] M. Gorfinkel. A decision-theory approach to missile defense. *Operations Research*, 11(2):199–209, 1963.

[41] Ho Y.C. Guan, X. and F. Lai. An ordinal optimization based bidding strategy for electric power suppliers in the daily energy market. *Power Systems, IEEE Transactions on*, 16(4):788–797, 2001.

[42] Y. Cheen H. Cai, J. Liu and H. Wang. Survey of the research on dynamic weapon-target assignment problem. *Journal of Systems Engineering and Electronics*, 17(3):559–565.

[43] S. Hanafi and A. Freville. An efficient tabu search approach for the 0–1 multidimensional knapsack problem. *European Journal of Operational Research*, 106(2):659–675, 1998.

[44] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7):733–750, 1998.

[45] T. Hegazy. Optimization of resource allocation and leveling using genetic algorithms. *Journal of construction engineering and management*, 125(3):167–175, 1999.

[46] Sreenivas R.S. Ho, Y.C. and P. Vakili. Ordinal optimization of deds. *Discrete Event Dynamic Systems*, 2(1):61–88, 1992.

[47] P.A. Hosein and M. Athans. Some analytical results for the dynamic weapon-target allocation problem. Technical report, DTIC Document, 1990.

[48] Walton J.T. Hosein, P.A. and M. Athans. Dynamic weapon-target assignment problems with vulnerable c^2 nodes. 1988.

[49] Fredrik Johansson and Göran Falkman. An empirical investigation of the static weapon-target allocation problem. In *Proceedings of the 3rd Skövde Workshop on Information Fusion Topics*, 2009.

[50] Fredrik Johansson and Goran Falkman. Real-time allocation of firing units to hostile targets. *Journal of Advances in Information Fusion*, 6(2):187–199, 2011.

[51] Orhan Karasakal. Air defense missile-target allocation models for a naval task group. *Computers & Operations Research*, 35(6):1759–1770, 2008.

[52] J.D. Katter. A solution of the multi-weapon, multi-target assignment problem. Working paper 26957, MITRE, 1986.

[53] Pferschy U. Kellerer, H. and D. Pisinger. *Knapsack Problems*. Springer Verlag, 2004.

[54] D. Khosla. Hybrid genetic approach for the dynamic weapon-target allocation problem. In *Proceedings of SPIE*, volume 4396, pages 248–263. SHE, 2001.

[55] Min Kong, Peng Tian, and Yucheng Kao. A new ant colony optimization algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 35(8):2672–2683, 2008.

[56] Cédric Leboucher, Hyo-Sang Shin, Patrick Siarry, Rachid Chelouah, Stéphane Le Ménec, and Antonios Tsourdos. *Recent Advances on Meta-Heuristics and Their Applications to Real Scenarios*, chapter A Two-Step Optimisation Method for Dynamic Weapon Target Assignment Problem. InTech, 2013.

[57] Lee C.Y. Lee, Z.J. and S.F. Su. An immunity-based ant colony optimization algorithm for solving weapon–target assignment problem. *Applied Soft Computing*, 2(1):39–47, 2002.

[58] Su S.F. Lee, Z.J. and C.Y. Lee. A genetic algorithm with domain knowledge for weapon-target assignment problems. *Journal of the Chinese Institute of Engineers*, 25(3):287–295, 2002.

[59] Su S.F. Lee, Z.J. and C.Y. Lee. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(1):113–121, 2003.

[60] S.P. Lloyd and H.S. Witsenhausen. Weapons allocation is np-complete. In *1986 Summer Computer Simulation Conference*, pages 1054–1058, 1986.

[61] Mian W. Lu, Y. and M. Li. The air defense missile optimum target assignment based on the improved genetic algorithm. *Journal of Theoretical and Applied Information Technology*, 48(2), 2013.

[62] K. Leblebicioğlu M.A. Sahin. Approximating the optimal mapping for weapon target assignment by fuzzy reasoning. *Information Sciences*, 2013.

[63] A.S. Manne. A target-assignment problem. *Operations Research*, 6(3):346–351, 1958.

[64] G. Manor and M. Kress. Optimality of the greedy shooting strategy in the presence of incomplete damage information. *Naval Research Logistics*, 44(7):613–622, 1997.

[65] Pisinger D. Martello, S. and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

[66] S. Matlin. A review of the literature on the missile-allocation problem. *Operations Research*, 18(2):334–373, 1970.

[67] Jau-yeu Menq, Pan-chio Tuan, and Ta-sheng Liu. Discrete markov ballistic missile defense system modeling. *European journal of operational research*, 178(2):560–578, 2007.

[68] William A Metler, Fred L Preston, and Jim Hofmann. A suite of weapon assignment algorithms for a sdi mid-course battle manager. Technical report, DTIC Document, 1990.

[69] R. Murphey. *Approximation and Complexity in Numerical Optimization Continuous and Discrete Problems*, volume 42, chapter An Approximate Algorithm for a Weapon Assignment Stochastic Program. Kluwer Academic, 2000.

[70] R.A. Murphey. An approximate algorithm for a weapon target assignment stochastic program. *Nonconvex Optimization and its Applications*, 42:406–421, 2000.

[71] R.A. Murphey. Target-based weapon target assignment problems. In *Nonlinear Assignment Problems*, pages 39–53. Springer, 2000.

[72] Yu Z. Ma F. Ni, M. and X. Wu. A lagrange relaxation method for solving weapon-target assignment problem. *Mathematical Problems in Engineering*, 2011, 2011.

[73] C. Novoa and R. Storer. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515, 2009.

[74] D. Orlin. Optimal weapons allocation against layered defenses. *Naval Research Logistics (NRL)*, 34(5):605–617, 1987.

[75] D. Orlin. Optimal weapons allocation against layered defenses. *Naval Research Logistics*, 34(5):605–617, 1987.

[76] Ahner D.K. Robbins M.J. Parson, C.R. Approximate dynamic programming methods for a cooperative dynamic weapon-target assignment problem. *Working paper*, 2014.

[77] Logan D.A. Patek, S.D. and D.A. Castanon. Approximate dynamic programming for the solution of multiplatform path planning problems. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 1061–1066. IEEE, 1999.

[78] TE Phipps and AL Karp. Optimum allocation of effort for deterrence. Technical report, DTIC Document, 1962.

[79] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

[80] Shapiro J.A. Powell, W.B. and H.P. Simão. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science*, 36(2):231–249, 2002.

[81] Warren B Powell, A George, B Bouzaiene-Ayari, and HP Simao. Approximate dynamic programming for high dimensional resource allocation problems. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 2989–2994. IEEE, 2003.

[82] W.B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[83] W.B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. Wiley-Interscience, 2007.

[84] W.B. Powell and B. Van Roy. Approximate dynamic programming for high dimensional resource allocation problems. *Handbook of learning and approximate dynamic programming*, pages 261–280, 2004.

[85] Hwang H.S. Pallerla R.P. Yucel A. Wilson R.L. Rosenberger, J.M. and E.G. Brungardt. The generalized weapon target assignment problem. In 10^{th} *International Command and Control Research and Technology Symposium: The Future of C2*, 2005.

[86] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27(11):1201–1225, 2000.

[87] Nicola Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[88] Gao Shang. Solving weapon-target assignment problems by a new ant colony algorithm. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, volume 1, pages 221–224. IEEE, 2008.

[89] Topi Sikanen. Solving weapon target assignment problem with dynamic programming. Technical report, Tech. Rep. 55670, 2008.

[90] S.N. Sivanandam and S.N. Deepa. *Genetic Algorithm Optimization Problems*. Springer, 2008.

[91] Richard M Soland. Optimal terminal defense tactics when several sequential engagements are possible. *Operations Research*, 35(4):537–542, 1987.

[92] C. Solnon and K. Ghédira. Ant colony optimization for multi-objective optimization problems. *International Journal on Computer Science*, 2010.

[93] Anabela P Tereso, M Madalena T Araújo, and Salah E Elmaghraby. Adaptive resource allocation in multimodal activity networks. *International Journal of Production Economics*, 92(1):1–10, 2004.

[94] Huseyin Topaloglu and Warren B Powell. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1):31–42, 2006.

[95] Ayes Turan. Algorithms for the weapon-target allocation problem. *Masters Thesis*, 2012.

[96] E. Wacholder. A neural network-based optimization algorithm for the static weapon-target assignment problem. *ORSA Journal on computing*, 1(4):232–246, 1989.

[97] Jun Wang, Xiaoguang Gao, and Yongwen Zhu. Solving algorithm for ta optimization model based on aco-sa. *Systems Engineering and Electronics, Journal of*, 22(4):628–639, 2011.

[98] Dean A Wilkening. A simple model for calculating ballistic missile defense effectiveness. *Science & Global Security*, 8(2):183–215, 2000.

[99] Wang H. Lu F. Wu, L. and P. Jia. An anytime algorithm based on modified ga for dynamic weapon-target allocation problem. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2020–2025. IEEE, 2008.

[100] Zhong J. Xie, M. and F.F. Wu. Multiyear transmission expansion planning using ordinal optimization. *Power Systems, IEEE Transactions on*, 22(4):1420–1428, 2007.

[101] Bin Xin, Jie Chen, Zhihong Peng, Lihua Dou, and Juan Zhang. An efficient rule-based constructive heuristic to solve dynamic weapon-target assignment problem. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(3):598–606, 2011.

[102] Bin Xin, Jie Chen, Juan Zhang, Lihua Dou, and Zhihong Peng. Efficient decision makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):649–662, 2010.

[103] Wang Yanxia, Qian Longjun, Guo Zhi, and Ma Lifeng. Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm. *Journal of Systems Engineering and Electronics*, 19(5):939–944, 2008.

[104] P.Y. Yin and J.Y. Wang. A particle swarm optimization approach to the nonlinear resource allocation problem. *Applied Mathematics and Computation*, 183(1):232–242, 2006.

[105] K.A. Yost and A.R. Washburn. Optimizing assignment of air-to-ground assets and bda sensors. *Military Operations Research*, 5(2):77–91, 2000.

[106] Liao Z. Wang J. Jiang B. Zhou, L. and Y. Yang. Hydrogen sulfide removal process embedded optimization of hydrogen network. *International Journal of Hydrogen Energy*, 37(23):18163–18174, 2012.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
26-12-2014	Dissertation	Oct 2011 - Dec 2014

4. TITLE AND SUBTITLE

Approximate Dynamic Programming for Military Resource Allocation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Parson, Carl, R.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-DS-14-D-16

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RW
Capt Melanie Walton
101 West Eglin Blvd, Bldg 13 Rm 233
Eglin AFB, FL 32542
(850) 882-3231

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RW

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A. Approved for Public Release;Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research considers the optimal allocation of weapons to a collection of targets with the objective of maximizing the value of destroyed targets. The weapon-target assignment (WTA) problem is a classic non-linear combinatorial optimization problem with an extensive history in operations research literature. The dynamic weapon target assignment (DWTA) problem aims to assign weapons optimally over time using the information gained to improve the outcome of their engagements. This research investigates various formulations of the DWTA problem and develops algorithms for their solution. Finally, an embedded optimization problem is introduced in which optimization of the multi-stage DWTA is used to determine optimal weaponeering of aircraft. Approximate dynamic programming is applied to the various formulations of the WTA problem. Like many in the field of combinatorial optimization, the DWTA problem suffers from the curses of dimensionality and exact solutions are often computationally intractability. As such, approximations are developed which exploit the special structure of the problem and allow for efficient convergence to high-quality local optima. Finally, a genetic algorithm solution framework is developed to test the embedded

15. SUBJECT TERMS

Weapon-Target Assignment, Dynamic Programming, Stochastic Programming, Stochastic Optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Darryl K. Ahner, AFIT/ENS
U	U	U	UU	173	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636 x4708 darryl.ahner@afit.edu

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18